Salesforce.com: Salesforce Summer '09

Development Lifecycle Guide

Enterprise Development on the Force.com Platform

Last updated: July 6, 2009

Table of Contents

Chapter 1

Introduction to the Force.com Development Lifecycle

Developing applications on the Force.com platform is easy, straightforward, and highly productive. A developer can define application components such as custom objects and fields, workflow rules, Visualforce pages, and Apex classes and triggers using the point-and-click tools of the Web interface. Simple changes can be implemented immediately without affecting other users in the production organization. More complex features can be left in development status until they have been fully tested, and then can be deployed so that everyone in the production organization may benefit.

But how do those same development practices work when building a large-scale *enterprise application* in collaboration with several other developers? When developing complex applications with highly customized logic and user interfaces, configuring on-the-fly in a single environment no longer makes sense. These types of applications take time to develop and require more formal practices to ensure they work as intended and meet users' needs. While the point-and-click Web user interface is easy to use, it is not designed to support enterprise application development, and there are some limitations with respect to traditional development practices.

Traditional application development typically requires a *development environment*—a place where you can develop code without affecting end users. Such a development environment typically includes high-performance development tools with features such as automated testing and version control to support team development. In order for this kind of development to occur on the Force.com platform, individual application components have been exposed as simple text-based files that can be transported, versioned, and merged. Traditional *project-based* development opens up new possibilities for Force.com development, but also requires new processes for development, migration, and synchronization.

Whether you are an architect, administrator, developer, or manager, this guide prepares you to undertake the development and release of complex applications on the Force.com platform.

Software as a Service Development

Traditional software development follows a standard process of development, testing, and deployment. Development occurs on desktop computers that share and version code through a source code control system (SCCS). The code is tested in a QA build environment, and after passing tests, gets assembled into a deployment package, which is then deployed to production. This familiar process is depicted in the following image:

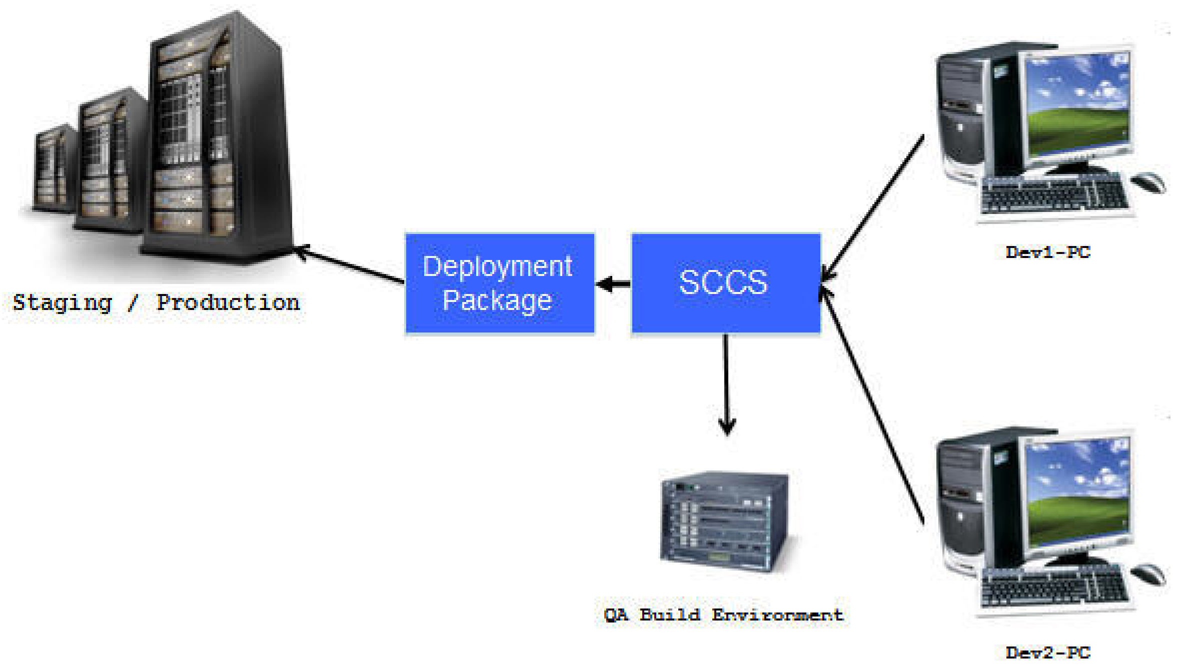


Figure 1: Traditional Software Development

Developing a cloud-based application is similar in many respects, with the main difference that most of the pieces are in the cloud. In the following image, notice that every cloud is itself a Salesforce organization.

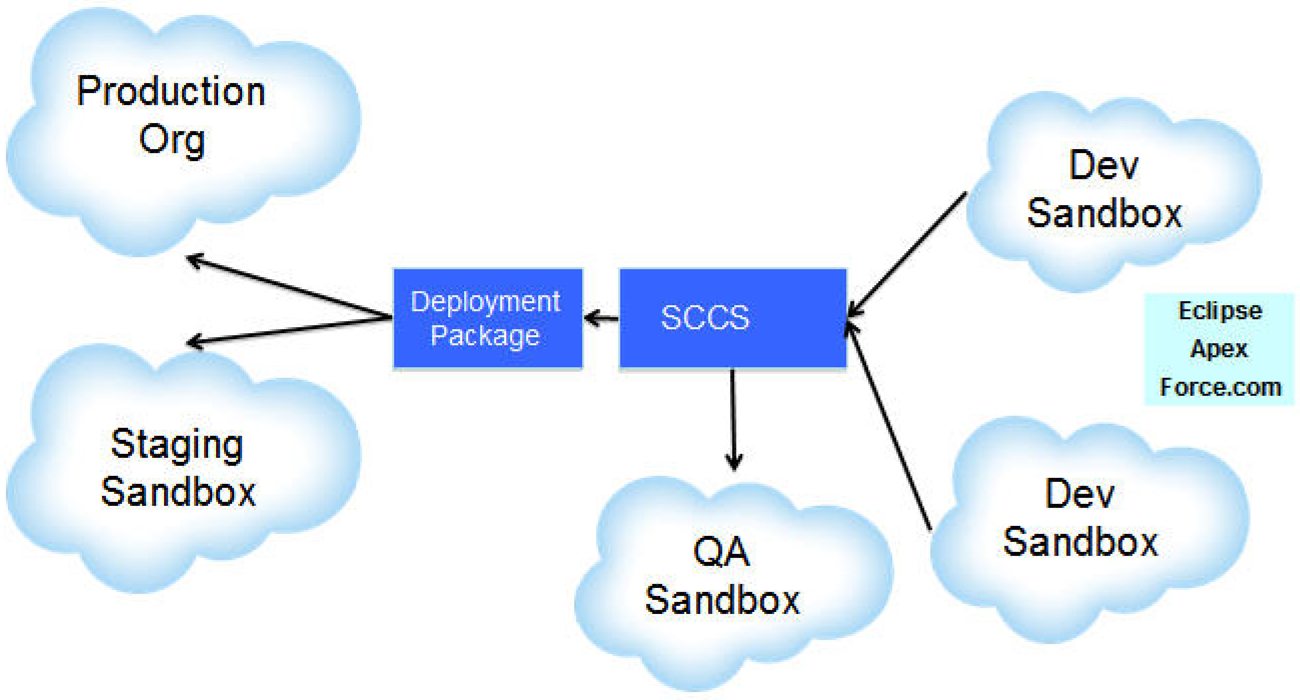


Figure 2: Cloud-based Software Development

The Force.com platform includes a suite of features that allows you to use traditional development practices for creating enterprise applications. These features can be divided into four areas:

Development environments

Having an isolated development environment keeps your production organization safe from changes that are in development. You build and test in a development environment before releasing new functionality to a production organization. There are two kinds of development environments: sandboxes and Developer Edition organizations.

File-based development

The Metadata API makes your organization's metadata downloadable as text-based files—you can retrieve components into a local project, modify the components, and then deploy to any organization.

Developer productivity tools

The Force.com platform contains a number of tools for creating applications, including the Force.com IDE, an integrated development environment that features code assist, integrated testing, wizards, and full platform documentation.

Migration tools

The Force.com platform provides the Force.com Migration Tool and the Force.com IDE for retrieving and deploying metadata. With these tools you can migrate changes between development environments, as well as deploy to production.

Who is this Guide For?

People developing on the Force.com platform generally are from either information technology (IT) or independent software vendors (ISVs). IT teams are concerned with the organization they are supporting; they want to make Salesforce work better for their users. ISVs, on the other hand, develop on the Force.com platform because they want to provide software as a service on the very best cloud computing platform. This guide is intended primarily for the former group, information technology. If you are an ISV looking to develop applications for the AppExchange, you will still find much of this guide useful. However, you will also want to see the topics in Additional Resources on page 6.

While the information in this guide is relevant to anyone creating or customizing applications on the Force.com platform, there are three main roles who will find this guide useful:

- Architects and Managers - Because service-based customizations affect the production organization immediately, architects need to be able to make decisions that control how application upgrades and modifications take place. What is the best way to develop new features without affecting the current application? How and when will you deliver the functionality? What is the best way to use your IT resources for development? Start with the following topics:
  - Application Lifecycle and Release Management on page 8
  - Scheduling Concurrent Development Projects on page 14
  - Development Environments on page 18
  - Establishing a Change Process for Production on page 14
  - Testing on page 47
  - Deploying to Production on page 51
  - Scheduling the Release on page 52
- Administrators - Administrators can make setup changes in the production organization that may affect how projects in development are deployed. Therefore, administrators play a vital role in how new technology is rolled out. Because administrators are crucial to the final deployment process, they should review the following topics:
  - Force.com Project-based Development on page 24
  - Development Environments on page 18
  - Establishing a Change Process for Production on page 14

- Development Tools on page 28
- Migrating Changes between Environments on page 39
- Deployment Dependencies on page 42
- Migrating Changes Manually on page 39
- Deploying to Production on page 51

- IT Developers - A developer needs to know the capabilities of the tools and how to implement the desired functionality. If you are a developer, quickly review the topics listed for managers, and then get working with the tools and technology. See the following topics for more information:
  - Force.com Project-based Development on page 24
  - Development Environments and Tools on page 18
  - Tracking Development Changes on page 33
  - Development Tools on page 28
  - Migrating Changes between Environments on page 39
  - Testing on page 47

Note: Application development on the Force.com platform is flexible, and while this guide has certain sections that are aimed specifically at one role or another, there is a lot of crossover. You may find that as an administrator, some of the topics for developers are important to you. Likewise, as a developer, there are portions of this guide aimed specifically at administrators or managers that can also be useful.

How is this Guide Organized?

The first two chapters of this guide lay the foundation for application development processes, while the remaining chapters describe the order in which you develop a Force.com application. This logical organization allows people of different roles to quickly orient themselves in the application lifecycle, and return to this guide later as a reference. This guide does not contain specific procedural instructions, for example, how to edit a project manifest or create a package. These details are documented in developer guides specific to the tool or technology in question. For more information, see Additional Resources on page 6.

- Introduction to the Force.com Development Lifecycle—describes cloud development, including the differences between Force.com development and traditional software development.
- Application Lifecycle and Release Management—examines release management processes, illustrated by various development scenarios of increasing complexity.
- Development Environments and Tools— begins with creating and configuring your development environment, followed by an introduction to Force.com project-based development, and then takes a brief look at the tools used for developing functionality and tracking changes.
- Migrating Changes between Environments—describes how to move changes between organizations, including how to determine which files to deploy in batches, and special considerations for deleting and renaming files in a deployment.
- Testing—details the steps required to create and configure a test environment for integration, regression, and staging.
- Deploying to Production—covers the high-level deployment procedure and best practices.
- Glossary—contains an alphabetical list of terms pertinent to this guide.

Additional Resources

Use the following sources to find more information about developing on the Force.com platform.

- Salesforce online help—Click the **Help & Training** link in the Web user interface for access to the Salesforce online help, tip sheets, and user guides.
- Developer documentation—The Force.com developer reference documentation is available from http://wiki.developerforce.com/index.php/Documentation.
- AppExchange Packaging Guide—This guide provides detailed information about packaging apps for the AppExchange. It is available at http://na1.salesforce.com/help/doc/en/salesforce_appexchange_publish_guide.pdf.
- Developer forums—The Force.com discussion boards are a great resource for developers helping developers. The forums are available from http://community.salesforce.com/sforce/.
- Developer Force—The Developer Force wiki at http://developer.force.com includes all the developer resources you need including technical articles, getting started tutorials, tools, the Force.com blog, and more.
- Force.com IDE—Click **Help** in the Force.com IDE to access the Force.com library.

Chapter 2

Application Lifecycle and Release Management

Force.com application lifecycle release management is similar to a jigsaw puzzle except that some of the pieces can change position over time. Therefore, knowing how all the pieces fit together is not enough; you must know how each piece moves during the development cycle. The duration and complexity of your application development lifecycle depends on several factors, but most important are the time required to develop the functionality and the specific processes of your IT department.

One of the most productive features of the Force.com platform is that customizations can be made directly in the Web user interface and become available to end users immediately. These kinds of changes do not require elaborate development tools or processes. From a release management standpoint, very little effort is required.

A more involved upgrade, such as creating a new user interface for an existing application, could require an environment where you can isolate your development efforts from production users. Reintegrating the changes you make in your development environment back into the production organization introduces a new level of complexity to the development process, because your production organization may have changed during that same time. Release management at this level requires tracking changes between the various environments to make sure there are no clashes when you deploy.

A complex application that affects all users might require multiple development and testing environments. This kind of long-term project can be further complicated by having concurrent projects on a different development cycles, and could require several integrations before the application is deployed to production. In many cases, all of these development efforts happen simultaneously. Release management at this level is a very complex puzzle.

How do you manage these development projects so they do not conflict with each other? How do you juggle the different time frames and production lifecycles? And how do you merge the changes between environments so there are no conflicts when you roll out the functionality in a production organization? The answers to these questions are in the way you construct your application lifecycle management process, and only by understanding all the variables can you make those decisions.

Application Delivery Strategy

The flexibility of the Force.com platform allows you to develop applications in different environments on different schedules. The options are numerous, and without some guidance, can be a source of confusion. This section explores some of the common development scenarios, from simple to complex, including the development steps and the types of features that can be developed under each scenario.

The following development scenarios assume that the release process is dependent on the complexity of the application being developed. However, some IT departments have a rigid process for making even the smallest change. If this is the case for your IT department, do not think of the complexity of the change as being relevant, instead review the development scenarios for ways you might implement changes to your organization.

Developing in a Production Organization

The easiest way to develop new functionality is to customize a production organization using the Salesforce Web user interface. You can develop new objects and applications that contain sophisticated components using declarative tools that are powerful and easy to use. In this scenario, all of the development occurs on the production organization, so there is no need for separate development or testing environments. While this process is the fastest way to complete a circuit of the application lifecycle, you are limited in what you can accomplish.

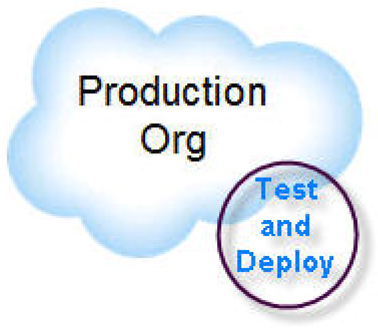


Figure 3: Production Development

Typical application lifecycle:

1. Plan functional requirements.
2. Develop using Salesforce Web tools.
3. Notify end users of changes.

Typical development projects:

- New custom fields on existing standard or custom objects
- New dashboards, reports, email templates
- New profiles

Developing with Sandbox

For development tasks that are slightly more complex or that must be isolated from the production organization, you can use a separate development environment, usually a sandbox. In this scenario, all of the development and testing occurs in the development environment, and then the changes are promoted to the production organization.

If you are simultaneously developing projects in production and sandbox organizations, you should considertracking setup changes you make in the production organization and replicating them in the sandbox. This is important because if your sandbox has out-of-date customizations, you could inadvertently replace your newer changes in production with these older customizations when you promote them from sandbox.

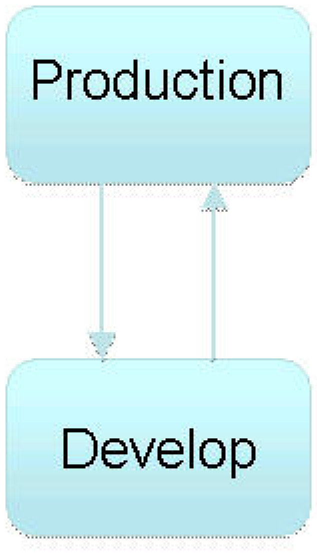


Figure 4: Single Development Environment

Typical application lifecycle:

1. Create a development environment.
2. Develop using Salesforce Web and local tools.
3. Test within the development environment.
4. Replicate production changes in development environment.
5. Schedule the release.

Typical development projects:

- New custom objects, tabs, and applications
- Integrations with other systems
- Apps involving Apex, Visualforce, workflow or new validation rules

Isolating Development and Testing

When using a single development environment for development and testing, you must stop development in order to test, and you can only resume development after you deploy. Unless you have a single developer doing all of these tasks, this can be an inefficient use of resources. A more sophisticated development model allows development to continue while testing and deployment take place. In this case you need two isolated environments, one for development and one for testing.

Having separate development and test environments increases the complexity of the development process, and introduces a question: where do changes to existing components take place? For example, imagine that you have an application that you have developed and you migrate those changes to your testing environment. During testing, you find that some changes need to take place in order to deploy to production. Do you make those changes in the testing environment or go back to your development organization and start the process again? If you only have two sandboxes, you may want to make those changes in your testing organization, because this is a faster and easier process, and your development sandbox may have already changed to the point where you cannot easily start over. However, you will still want to replicate any changes made in your test environment back to your development environment, to ensure the next time you promote from development to test that the fix is retained. The arrangement and flow of the application development process is shown in the following image:

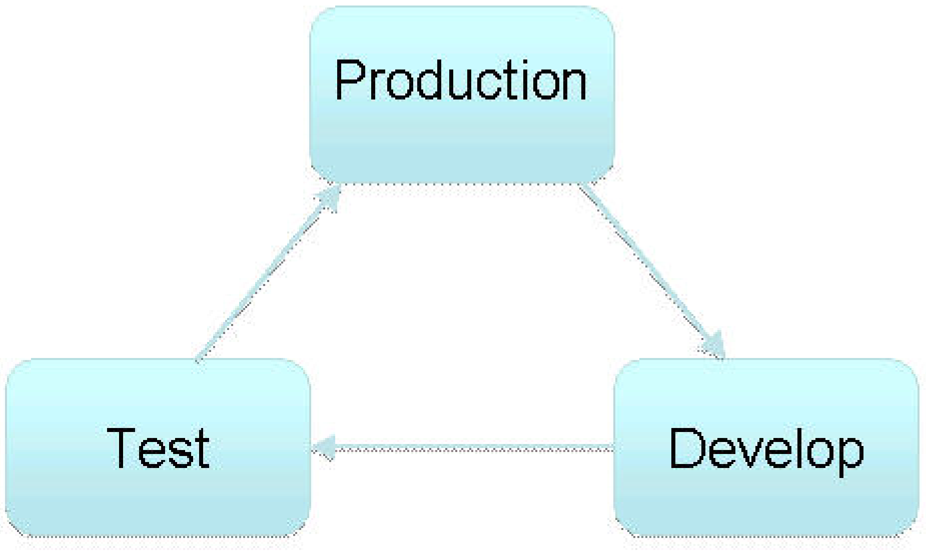


Figure 5: Development with Isolated Testing Environment

Typical development lifecycle:

1. Create a development environment.
2. Develop using Salesforce Web and local tools.
3. Create a testing environment.
4. Migrate changes from development environment to testing environment.
5. Test.
6. Replicate production changes in development environment.
7. Schedule the release.

Typical development projects:

- New custom objects, tabs, and applications
- Integrations with other systems
- Apps involving workflow or new validation rules

Developing Multiple Projects with Integration, Testing, and Staging

If you have multiple projects under development that will be released at the same time, you need *integration testing* to make sure that the separate code lines can merged without conflict. In the future, you might want to include *user-acceptance testing* (UAT) to determine if the original design requirements are met. A more elaborate development process could also contain a staging environment where you can ensure that the deployment to production will go exactly as planned.

In such a development scenario, you might wonder where you should make changes to the code line. If a feature fails any test on the way towards production deployment, do you fix the feature in the organization where it failed, or start the entire process over again? As the complexity of your application development process increases, you will find that fixing things as you go is not a good model. Instead, you will want to make all fixes in the development organization where the feature started and follow a repeatable process for migrating that change toward production. The following image depicts this process:

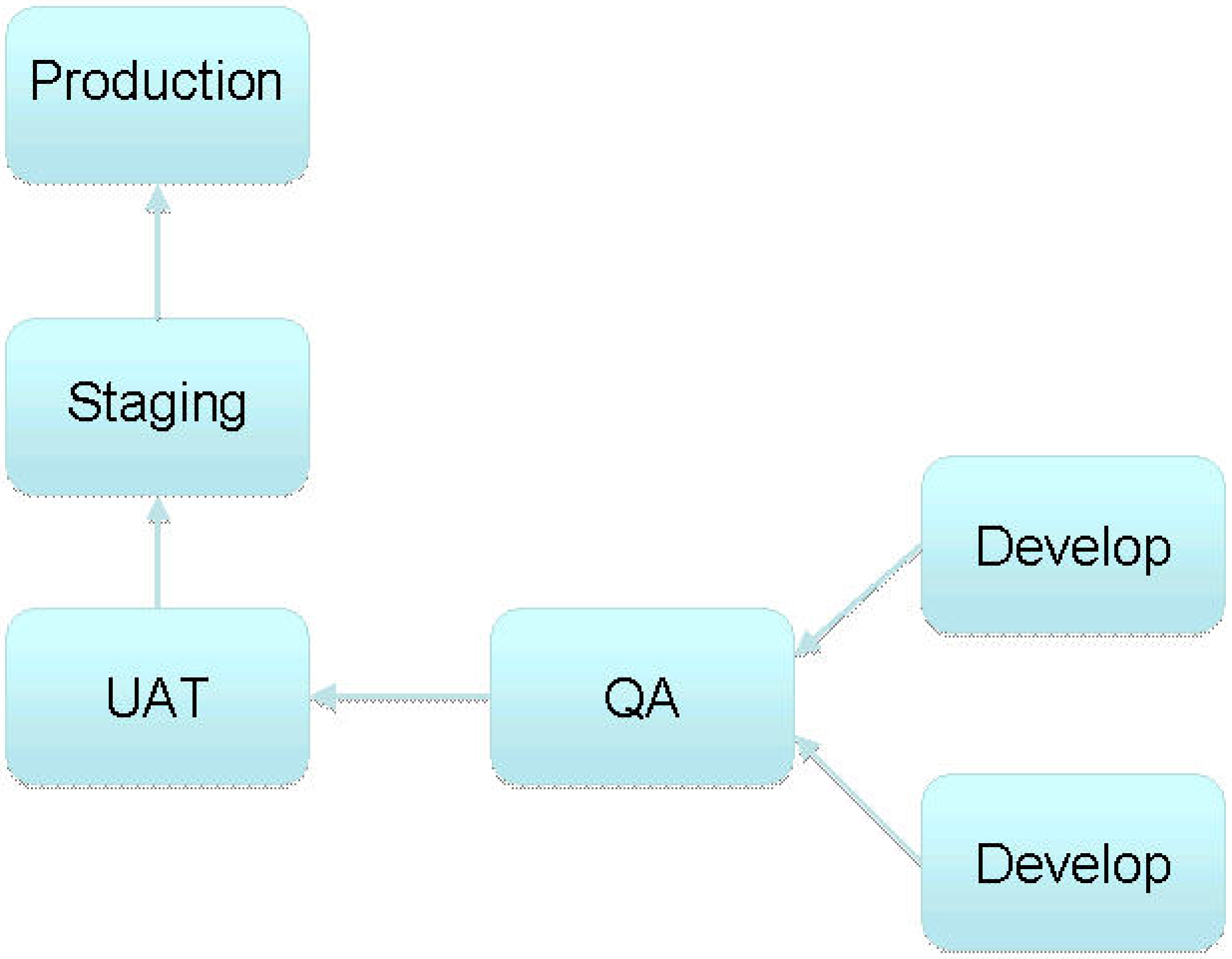


Figure 6: Multiple Development and Testing Environments

Typical development lifecycle:

1. Create development environments.
2. Develop using Salesforce Web and local tools.
3. Create testing environments, including UAT and integration.
4. Migrate changes from development environment to integration environment.
5. Test.
6. Migrate changes from integration environment to UAT environment.
7. Perform user-acceptance tests.
8. Migrate changes from UAT environment to staging environment.
9. Replicate production changes in staging environment.
10. Schedule the release.

Typical development projects:

- Concurrent development of new applications in multiple environments
- Projects that require team development

Developing Enterprise Applications

Large organizations tend to have complex development processes that span multiple release schedules. In this case, it is not only the division between development and testing that is important, but the synchronization of projects on different schedules. In this development scenario, you have multiple development environments that must integrate with each other before merging into a staging area. Additional environments could be added for training, production support, or other purposes. An organization can manage this type of enterprise development in a number of ways, one possible method is shown in the following image:

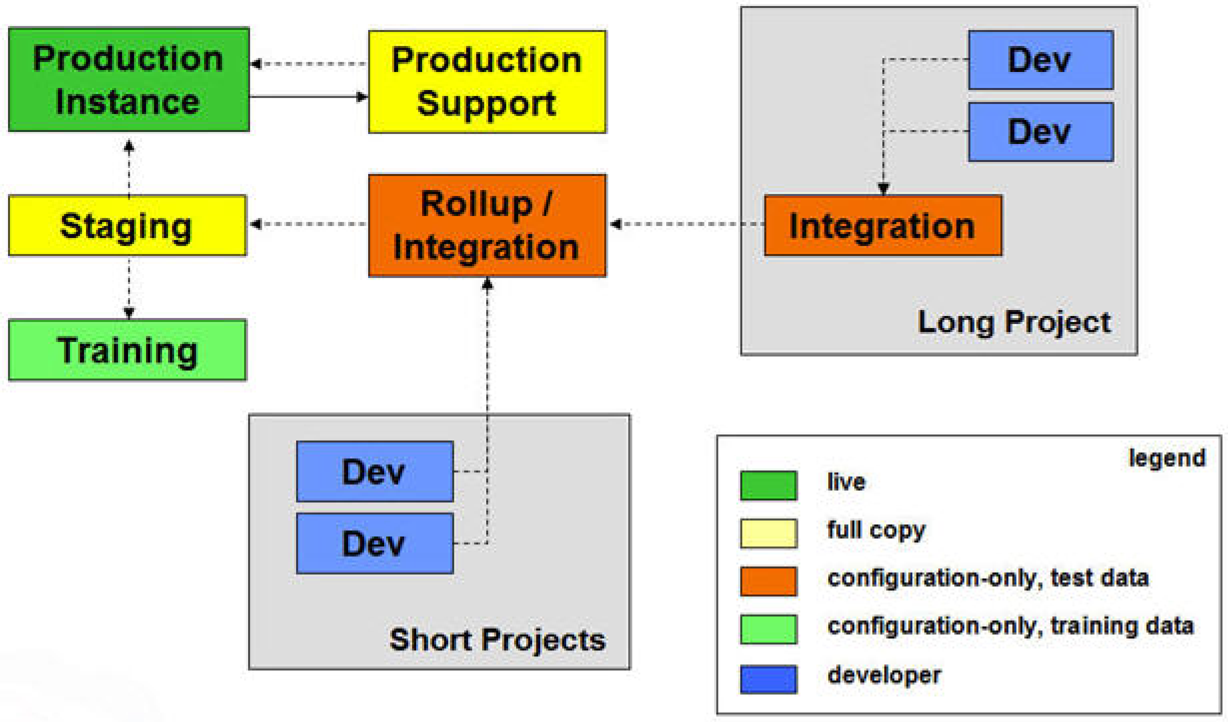


Figure 7: Multiple Development and Testing Environments

Typical development lifecycle:

1. Create development environments.
2. Develop using Salesforce Web and local tools.
3. Create testing environments, including UAT and integration.
4. Migrate changes from development environment to integration environment.
5. Test.
6. Migrate changes from integration environment to roll-up and UAT environments.
7. Perform user-acceptance tests.
8. Migrate changes from UAT environment to staging environment.
9. Replicate production changes in staging environment.
10. Migrate changes to training environment.
11. Schedule the release.

Typical development projects:

- Multiple projects of various complexities and durations on different release schedules
- Development teams that include distributed developers and testers

Establishing a Change Process for Production

The ability to quickly develop and customize applications in your production organization using the Salesforce Web user interface is one of the many strengths of the Force.com platform. However, when it comes to developing enterprise applications, this ease of development can lead to changes occurring on the production organization while applications are being developed in sandbox. In order to guarantee a successful deployment to your production organization, it is necessary for your development environments to have the same changes that occurred on production. This may seem like a backwards thing to do, to move modifications from your production organization to your development environments, but this is necessary, because migrating from development to production can overwrite the changes already made on production.

When modifications occur on the production organization, you can perform a sandbox refresh and get the latest changes. However, sandbox refreshes are not always possible (you are limited to one refresh every 29 days), and may require you to perform configuration changes, such as modifying user profiles so that your developers have the necessary permissions.

Therefore, tracking changes between the production and development environments, and then merging those differences from production to development, can be a necessary process. To make these tasks easier, it is a good idea to establish a change process for your production organization. A change process determines what kinds of modifications can take place on your production organization, when they can occur, and who is responsible for making the changes. The change process you adopt will depend on the kinds of modifications you require in your production environment. The following list suggests some best practices for change processes, arranged from simplest to most complex.

- Allow no changes on production—This is the simplest, but also the most draconian measure you can enforce. In effect, you are sacrificing immediate setup changes for easier deployment. If you choose this process, all of the development happens on sandbox.
- Modify only components in the Metadata API—Manually tracking and deploying changes is far more complicated than using the Metadata API. If you can limit production changes to components that are accessible through the Metadata API, then this will simplify change tracking, merging, and deployment.
- Allow only one administrator to make setup changes—Some organizations find it useful to assign a single administrator who is responsible for all setup changes on production. This makes it easier to track changes on the production organization and replicate those changes back into development environments between sandbox refreshes. This is a more flexible approach that allows changes in production and project-based development at the same time. However, this is only a workable solution if your organization is small enough that one administrator can make all of the setup changes.
- Schedule production changes—If your organization requires frequent changes to the production environment, you can schedule a time to migrate those changes to your development environments. Depending on the number of organizations you have, this could be an easy migration done frequently (weekly, perhaps), or a more elaborate process performed less often (once per quarter).

Scheduling Concurrent Development Projects

Unlike traditional software development, where all upgrades occur in a single release, an online application can be upgraded at any time. Enhancements that are easier to develop, require less testing, or are of higher priority can be released to end users very quickly. Therefore, an important step in release management is scheduling your development efforts. If application lifecycle management is like a jigsaw puzzle, then scheduling your development projects is like grouping the puzzle pieces: put the corners in place, collect the sides in one pile, and group the remainder by color. For IT organizations, this means putting development projects into categories such as short-term, medium-term, and long-term. These categories are often defined by where development takes place, how much testing is required, and when new features must be available.

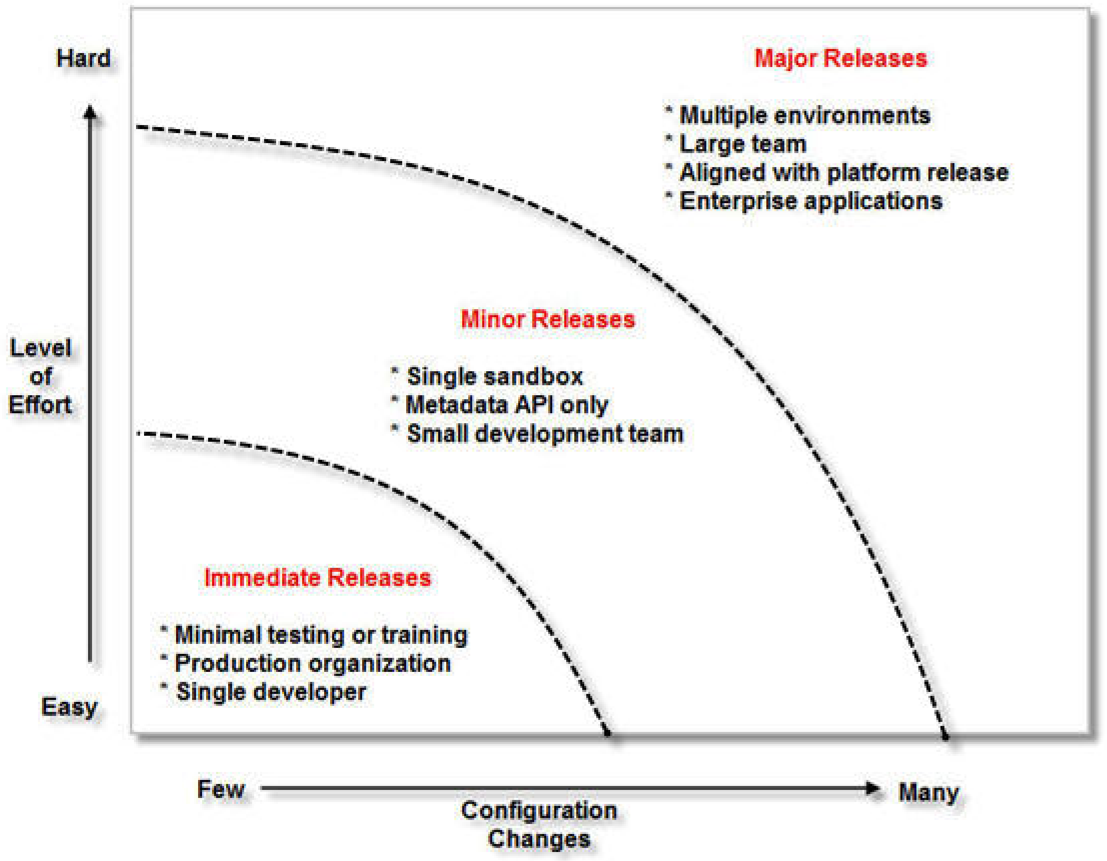


Figure 8: Release Management Strategy

While you can use as many categories as necessary, a three-tier scheme is a good starting point. Aside from the duration of the project, here are some other possible ways you can categorize your development efforts.

Where development takes place:

- Production-only—If the functionality can be developed and tested entirely in the production Web interface, the development cycle is faster and users can get the functionality sooner.
- Metadata API components—If all of the necessary components are available in the Metadata API, then it is much easier to track and merge changes between environments.
- Single sandbox—If the functionality can be developed in a sandbox and then immediately deployed to a production organization, the development cycle does not require integration or staging environments.
- Multiple environments—Development projects can span multiple sandboxes, in which case the complexity of integrating codelines is increased. Complicated projects should not keep the simple ones from being rolled out.

By the number of developers:

- One—If a single developer can create, test, and deploy the functionality, you are far less likely to run into problems merging changes, or other time-consuming issues.
- Small team—A small development team can partition large projects into manageable pieces, and is still capable of rapid development. Projects of this nature can be easily integrated with single-developer projects and rolled out quickly.
- Large team—A full development team is necessary for large-scale development projects. Projects of this nature require tracking and merging changes from different code branches, acceptance testing, and other involved processes that can slow down the development process.

For more information on the projects you might develop in each category and the general development process, see Application Delivery Strategy on page 8.

Delivering Applications on a Release Train

A *release train* is a scheduling technique for delivering application upgrades on a regular basis. Release trains are predictable and incremental, so they ease the development process by setting limits on how much can be done in any one development cycle. Because updates to your application can be impacted by Salesforce upgrades, you might find it useful to schedule your release before Salesforce upgrades occur.

The general process for delivering multiple applications on a release train is the following:

1. Plan your release around Salesforce upgrades.
2. Schedule your concurrent development projects. This will help you determine if the new functionality can be done now, in the current release, or sometime in the future.
3. Establish a process for changes to the production organization.
4. Track changes in all environments. This will sure that short-term functionality that is delivered to production does not get overwritten when you deploy from your development environment.
5. Schedule the release.

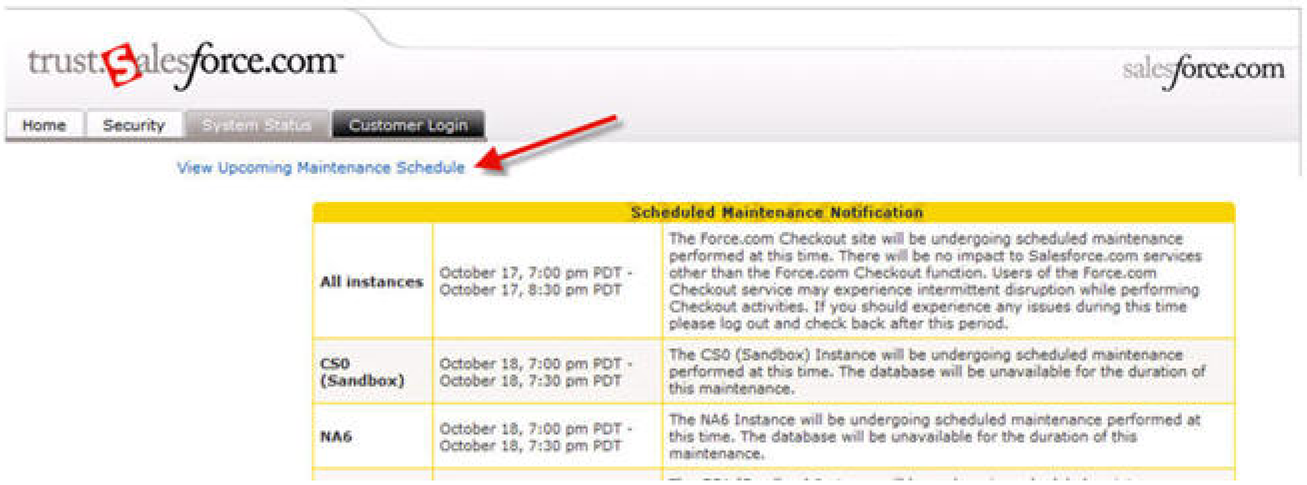
How Salesforce Upgrades May Affect Delivery Schedules

Several things happen when Salesforce upgrades to the next version; most importantly, the servers shut down. Upgrades are scheduled during off-peak hours, and rarely affect users. However, IT departments often schedule their own batch processes during these same hours, so it is important to avoid any conflicts. Other things that happen during an upgrade are:

- New logo—The Salesforce logo is a quick way to verify which version you are using. Sandboxes may upgrade before or after your production organization; therefore, it is a good idea to check the logo in the upper left corner of your sandbox home page around the time of a Salesforce release to see if the sandbox has upgraded or not.
- New features—Every release contains new features, among those are new components available through the Metadata API. Click the **What's New** link in the Help and Training window to view the release notes.
- Incremented API version—The API version increments every release, and access to new features requires connecting to the new version of the API.

Viewing the Upcoming Maintenance Schedule

You can view the upcoming maintenance schedule by visiting trust.salesforce.com and clicking the **View Upcoming Maintenance Schedule** link.



Scheduled Maintenance Notification		
All instances	October 17, 7:00 pm PDT - October 17, 8:30 pm PDT	The Force.com Checkout site will be undergoing scheduled maintenance performed at this time. There will be no impact to Salesforce.com services other than the Force.com Checkout function. Users of the Force.com Checkout service may experience intermittent disruption while performing Checkout activities. If you should experience any issues during this time please log out and check back after this period.
CS0 (Sandbox)	October 18, 7:00 pm PDT - October 18, 7:30 pm PDT	The CS0 (Sandbox) Instance will be undergoing scheduled maintenance performed at this time. The database will be unavailable for the duration of this maintenance.
NA6	October 18, 7:00 pm PDT - October 18, 7:30 pm PDT	The NA6 Instance will be undergoing scheduled maintenance performed at this time. The database will be unavailable for the duration of this maintenance.

Figure 9: Viewing Upcoming Maintenance Schedules

About Sandbox Upgrades

Sandbox and production instances upgrade at different times. This means that, for a brief time, your sandbox and production organizations are running different versions of Salesforce. A sandbox may be upgraded earlier or later than the production organization, depending on the instance. The schedule is determined by the sandbox copy date.

Chapter 3

Development Environments and Tools

Developing on the Force.com platform requires a *development environment*, a place where you can work without affecting other users. In traditional software development, a development environment is little more than a space to call your own, but on the Force.com platform, each development environment is its own fully functional Salesforce organization.

After you create and configure your development environment, you can begin to develop functionality using a mixture of traditional development tools and a powerful Web-based declarative interface. If you have multiple development environments, you need to merge your changes with changes from other team members into a single development environment. During this phase, you must track changes in all environments so that you can merge those changes successfully.

Development Environments

A *development environment* is a Salesforce organization where you can make configuration changes that will not affect users on the production Salesforce organization. Development environments are essential to enterprise application development, and most development organizations will have multiple environments for various purposes, such as splitting up development tasks, testing, and training.

There are two kinds of development environments: sandbox organizations and Developer Edition organizations.

About Sandbox Organizations

A sandbox is a copy of your production organization's setup configuration; in other words, a sandbox is a replica of all of the *metadata* in your production organization. Your production organization's configuration (the custom objects and fields, applications, workflow, and anything else you have created to describe your organization and business processes) is copied to an area that is completely isolated from your production organization. Operations you perform in your sandbox do not affect your production organization, and vice versa.

You can create multiple sandbox copies of your organization for a variety of purposes, without compromising the data and applications in your Salesforce production organization. For example, you might have one sandbox for development, another sandbox dedicated to testing, and a third sandbox for training. Each sandbox instance is isolated from every other sandbox instance, just as they are from production.

There are three types of sandboxes:

Configuration Only

Configuration-only sandboxes copy all of your production organization's reports, dashboards, price books, products, apps, and customizations under **Setup**, but exclude all of your organization's standard and custom object records, documents, and attachments. Creating a configuration-only sandbox can decrease the time it takes to create or refresh a sandbox from several hours to just a few minutes, but it can only include up to 500 MB of data.

Developer Sandbox

Developer Sandboxes are special configuration-only sandboxes intended for coding and testing by a single developer. They provide an environment in which changes under active development can be isolated until they are ready to be shared. Just like configuration-only sandboxes, Developer Sandboxes copy all application and configuration information to the sandbox. Developer Sandboxes are limited to 10 MB of test or sample data, which is enough for many development and testing tasks.

Full

Full sandboxes copy your entire production organization and all of its data, including standard and custom object records, documents, and attachments.

A sandbox may also contain some, none, or all of your production data, depending on the intended use: For development, you might only need a small set of data to make sure things work. For QA testing, especially regression testing, you need a large set of data that does not change over time. For pre-deployment staging, you need a sandbox that is as close to your production environment as possible, including the data. Throughout the application development lifecycle, enterprise development requires many sandboxes for different purposes. The following table lists the different kinds of sandboxes and which development role they are best suited for.

Table 1: Development Environments and Typical Uses

Use	Developer Sandbox	Configuration-only Sandbox	Full-copy Sandbox
Development	• Best for app development	• Good for app development	• Giving developers access to data may not be OK • Slower to copy
Testing	• Unit tests • Apex tests	• Best for feature tests • Load standard data for regression tests	• Best for production debugging
Testing external integrations	• Not a good fit	• Special cases only • Use sample or subset data • Works well if using external IDs	• Best when external system expects full production data to be present
Staging/User-acceptance Testing	• Not a good fit	• Sometimes appropriate if testing against a subset of production data is acceptable, for example, regional tests	• Best for validation of new apps against production configuration and data
Limitations	• 10 MB storage	• 500 MB storage	• Limited number available per organization • Duration to create and refresh

Production Features Disabled on Sandbox

Features that automatically send email to contacts, customers, and users are disabled in sandboxes, and cannot be enabled:

- Case escalation

- Opportunity reminders
- Contract expiration warnings
- Subscription summary
- Automated weekly data exports
- The ability to create Salesforce sandboxes.

  Case escalation, opportunity reminders, and contract expiration warnings are disabled because they automatically send email to contacts, customers and users who should not interact with sandboxes.
- Testing Salesforce Content in your sandbox is not supported.
- Testing Salesforce for Google AdWords in your sandbox is not supported. Attempting to test Salesforce for Google AdWords in your sandbox will result in errors because your sandbox organization operates with the same link to your Google AdWords account as your production organization.
- Email service addresses that you create in your sandbox cannot be copied to your production organization.

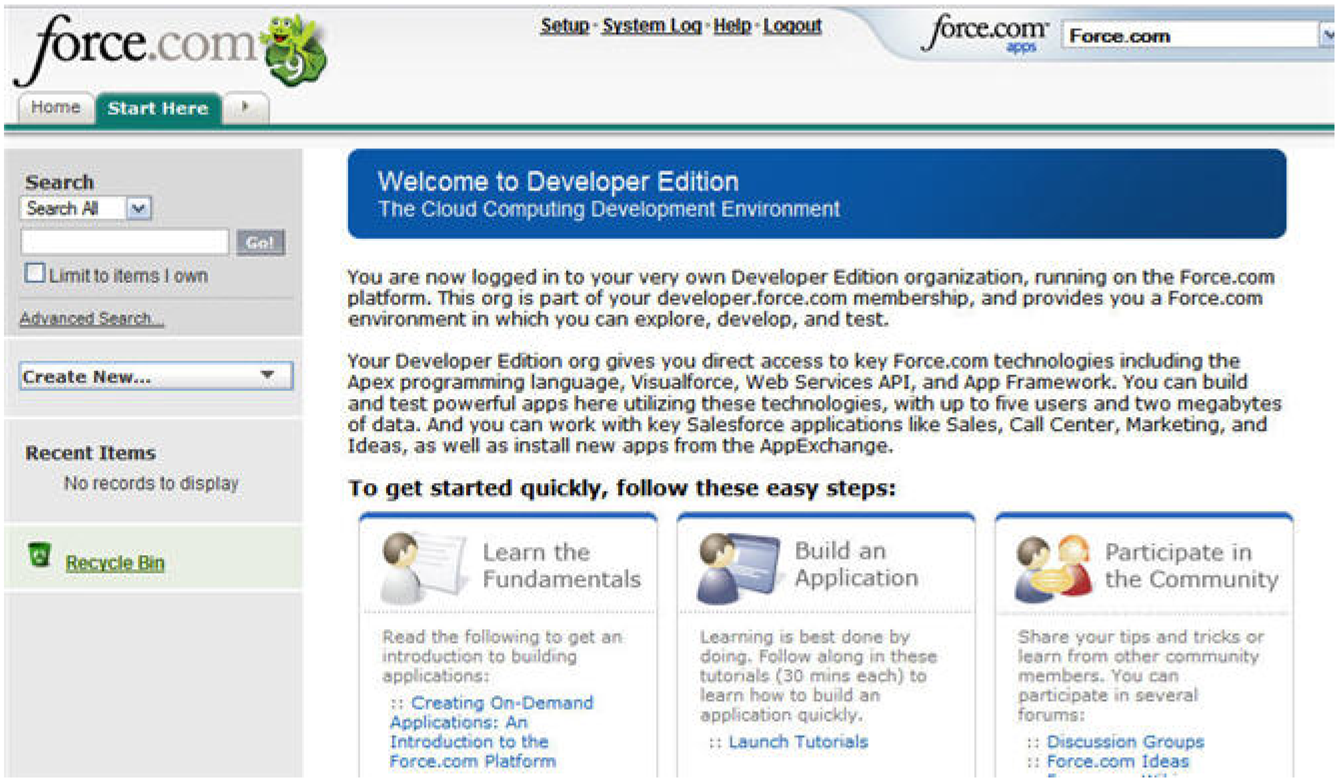
Considerations for Creating or Refreshing a Sandbox

Note the following important considerations when you create or refresh a sandbox:

- Plan sandbox refreshes well ahead of time. Creating or refreshing a sandbox copy is a long-running operation that may complete in minutes, days, or even more than a week. A number of conditions factor into the duration, including the number of customizations, data size (for full copies), numbers of objects, and server load. Also, sandbox refreshes are queued, so your requested copy may not start immediately after your request. You are notified of the completion of a sandbox copy via email.
- Refreshing a sandbox deletes and recreates the sandbox as a new copy of the production organization. In effect, this reverses any manual access changes you have performed. If you created users on sandbox, they will no longer exist; if you changed a user's profile and permissions, those will revert to their values in the production organization. This means that after a refresh, any access changes you performed must be repeated in the new copy. To avoid this process, you can create user templates in your production organization, and then activate them in the sandbox organization.
- Freeze all changes to your production organization while a sandbox is being created or refreshed. Setup and data changes to your production organization during the sandbox creation and refresh operations may result in inconsistencies in your sandbox.

About Developer Edition Organizations

A Developer Edition organization is primarily useful for independent software vendors (ISVs) who want to create new applications on the Force.com platform and package them for distribution on the AppExchange. However, Developer Edition organizations can also be used any time you require an organization for development purposes.



You are now logged in to your very own Developer Edition organization, running on the Force.com platform. This org is part of your developer.force.com membership, and provides you a Force.com environment in which you can explore, develop, and test.

Your Developer Edition org gives you direct access to key Force.com technologies including the Apex programming language, Visualforce, Web Services API, and App Framework. You can build and test powerful apps here utilizing these technologies, with up to five users and two megabytes of data. And you can work with key Salesforce applications like Sales, Call Center, Marketing, and Ideas, as well as install new apps from the AppExchange.

To get started quickly, follow these easy steps:

Learn the Fundamentals

Read the following to get an introduction to building applications:

:: Creating On-Demand Applications: An Introduction to the Force.com Platform

Build an Application

Learning is best done by doing. Follow along in these tutorials (30 mins each) to learn how to build an application quickly.

:: Launch Tutorials

Participate in the Community

Share your tips and tricks or learn from other community members. You can participate in several forums:

:: Discussion Groups
:: Force.com Ideas

Figure 10: Developer Edition

Developer Edition provides free access to many of the exclusive features available with Enterprise and Unlimited Editions. You get full access to the Force.com platform and API, so that you can extend Salesforce, integrate with other applications, and develop new tools and applications.

Note: Salesforce provides technical support for sandboxes, but does not provide support for Developer Edition organizations. Therefore, the level of technical assistance you require may dictate which type of organization you choose.

Developer Edition has a limited number of licenses:

- Two full CRM Licenses
- Three Salesforce user licenses
- Five Partner user licenses
- Ten Customer Portal Manager user licenses
- Two Marketing feature licenses
- Two Mobile feature licenses
- Two Offline feature licenses

Note: The licensing information listed is valid as of 09. If you signed up before November 2008, there were fewer licenses available. Licensing may also change in future releases. For additional information on Developer Edition, go to http://developer.force.com/.

Choosing a Development Environment

Use a sandbox as your development environment if:

- You have an Enterprise or Unlimited Edition Salesforce organization.

- You are creating functionality for a single Salesforce organization, because you can copy your entire organization's metadata to the sandbox in one click.
- You need to do testing on data that is similar to your production organization.
- You have had special features enabled, such as Person Accounts.

Use a Developer Edition organization if:

- You are developing functionality that does not depend on the rest of your organization. In most cases, it is probably still better to use a sandbox.
- You have used all of your allocated sandboxes already.
- You have a Professional, Group, or Personal Edition Salesforce organization, because sandboxes are not available to you.
- You are an independent software vendor (ISV) creating an application for distribution on the AppExchange. Note that this development scenario is not covered by this guide. For more information, see the topics listed in Additional Resources on page 6.

Note: You can create a development environment that is similar to a sandbox by retrieving all of your organization's metadata into a Force.com project, and then deploying the project to a Developer Edition organization. However, not all metadata types can be retrieved and deployed, and not all features in your production organization may be enabled in a Developer Edition organization, so your development environment will not be as complete of a replica as a sandbox.

Creating a Development Environment

After you have determined which kind of development environment you need for a particular project or task, start a sandbox copy or sign up for a Developer Edition organization.

To start a sandbox copy:

1. Click **Setup** ➤ **Data Management** ➤ **Sandbox**.
2. Click **New Sandbox**.

Note: The number and availability of sandboxes depends on the Salesforce organization. For more information, see Development Environments on page 18.

To sign up for a Developer Edition organization:

1. In a Web browser, navigate to developer.force.com.
2. Click the **Developer Login** tab.
3. Click **Sign up Now**.

Once your development environment is created, you may still need to configure it further to update environmental dependencies and merge project contents.

Updating Environmental Dependencies

Environmental dependencies are settings that are different between a development environment and the production organization. When you work in a development environment, you need to update who has access to what, turn certain features on or off, and update the addresses of internal and external systems so that they point to the development environment instead of production. The reverse is also true—when you deploy to production, you may need to update certain settings that you used in development so they work with production.

The following list outlines common environmental dependencies:

- **Login privileges**—If your developers and testers do not have logins on production, or do not have the necessary privileges, you need to give them the necessary access in the development environment.
- **Email addresses**—When you create a sandbox, email addresses are automatically changed so that email alerts and other automatic notifications are not sent from the sandbox during development. When your developers and testers log into the sandbox, they must change their email address back to a real address in order to receive email messages sent from the sandbox.
- **Email recipients**—If you want to test outbound email for features such as escalations or dashboards, you must update the list of recipients, because these lists are removed when the sandbox is created in order to prevent unnecessary emails.
- **External URLs**—If your production organization is integrated with external systems, you probably will not want your sandbox copy to communicate with the production versions of those external systems, so as not to mix production and non-production data. For example, if you use outbound messaging or web service callouts, you will want to update the URLs called by those services to point to your external development environment(s) for those applications. Likewise, since sandboxes run on a different instance than the production organization, to test integrations with sandbox you will need to change the Web Service API endpoint hostname from `www.salesforce.com` to `test.salesforce.com`.
- **Hard-coded URLs**—In general, when linking from one Force.com page to another, the links should be relative (omitting the hostname) rather than absolute, and dynamic (looking up the ID a record or page by name) rather than static. This allows you to migrate the URLs between different organizations, which may have different hostnames or record IDs. If your application contains hard-coded URLs from one Force.com page to another, they may break when they are clicked from the sandbox or, worse, take a user who thinks he is in a development environment back to the production organization.
- **Hard-coded IDs**—Some components are commonly accessed via their ID, such as RecordTypes. When you create a sandbox copy, code that uses these IDs to reference these components continues to work because production IDs are retained by the sandbox copy. However, the reverse is not true--migrating from sandbox to production (or between any two organizations) via metadata maintains components' names, but if the component does not exist in the target organization a new ID will be generated. Therefore, migrating code that contains a hard-coded ID can cause the code to break. As a best practice, wherever possible you should obtain a component's ID by querying for it by name.
- **Existing projects**—If you have existing development projects, you need to merge that work into the new development environment. For more information, see Migrating Changes between Environments on page 39.

Creating User Templates

When you refresh a sandbox, a new copy of the production organization is created. This means that all user permissions within the sandbox will be copied from production, and that user access or permissions changes you made in the sandbox before it was refreshed must be reapplied. If you have multiple sandboxes, or refresh sandboxes on a regular basis, consider creating a developer user template on your production organization. A developer user template is an abstract user that has permissions required to develop on the sandbox (for example, the "Modify All Data" permission), but is not active on your production organization. After you create or refresh a sandbox, you activate the developer user in the sandbox and assign it to the individual who will be developing there.

To create a developer template:

1. Create a new user on your production organization.
2. Edit the user to give it the necessary permissions.
3. Deactivate the user on the production organization.
4. Create or refresh a sandbox.
5. Activate the user on the sandbox.
6. Optionally change the email address, password, or other environmental settings.

Note: Exercise caution when granting permissions in sandbox organizations that contain sensitive information copied from production (for example, social security numbers).

You might find it helpful to create multiple templates for different roles. For example, you could have a developer role that has the "Modify All Data" permission, and a QA tester role that has standard user privileges.

Your sandbox has the same number of licenses as production, but it is unlikely that all of your users will log into it. When you create or refresh a sandbox, the same users who are active in production are active in the sandbox, so if all licenses are occupied you will need to deactivate an active user in order to activate the inactive developer user. Just make sure that the user you're deactivating in sandbox is one of the many production users who will never log into that environment. Likewise, you should make your production developer user template inactive on the production organization, so it doesn't consume a paid production license.

For more information on how to activate, deactivate, or edit users, see "Editing Users" in the Salesforce online help.

Managing Sandboxes

To manage your sandboxes, click **Setup ➤ Data Management ➤ Sandbox**. A list of your existing sandboxes displays.

- Click **New Sandbox** to create a new sandbox. For information on different kinds of sandboxes, see About Sandbox Organizations on page 18.

  Salesforce deactivates the **New Sandbox** button when an organization reaches its sandbox limit. If necessary, contact salesforce.com to order more sandboxes for your organization.

  Note that Salesforce deactivates all refresh links if you have exceeded your sandbox limit.
- Click **Show Sandbox Refreshes** to see a log of your sandbox refresh history, including when sandboxes were created and who created them.
- Click **Refresh** to replace an existing sandbox with a new copy. Salesforce only displays the **Refresh** link for sandboxes that are eligible for refreshing, which is any time after 30 days from the previous creation or refresh of that sandbox. Your existing copy of this sandbox remains available while you wait for the refresh to complete. The refreshed copy is inactive until you activate it.
- Click **Activate** to activate a refreshed sandbox. You must activate your refreshed sandbox before you can access it. Salesforce only displays this option for sandboxes that are not activated.

  Caution: Activating a refreshed sandbox replaces the existing sandbox with the refreshed version. This permanently deletes the existing version and all data in it. Your production organization and its data will not be affected.

- Click **Del** to delete a sandbox. The delete option is only available if your organization exceeds its sandbox limit.

  Caution: Deleting a sandbox permanently erases the sandbox and all data in it.

- Click **Login** to log in to a sandbox. Salesforce only displays this option for active sandboxes.

  Note that the **Login** button is for administrators and may not always be available; however, you can log into an active sandbox at `https://test.salesforce.com` by entering your modified username and password.
- Click on a sandbox name to view details about the sandbox, including the sandbox type and when it was created.

Force.com Project-based Development

Force.com *project-based development* leverages the tools and processes of traditional software development for online development. Integrated development environments (IDEs), source control systems, and development environments are familiar to most traditional software developers, but may be foreign to Salesforce administrators. However, these tools are essential for project-based development.

The Force.com platform enables project-based development by using text-based files to represent the various components in a Salesforce organization. These files are easily transported, can be stored and versioned in a source control system, and enable traditional development. All of this is made possible by the Metadata API.

What is the Metadata API?

The Metadata API provides two parts that work in conjunction: a rich and powerful metadata model and an application programming interface (API). The metadata model describes the components in your organization. For example, Salesforce custom objects and their fields can be controlled by manipulating the XML files provided by the Metadata API. The Metadata API also includes a set of Web service methods that provide programmatic access to the configuration and source of your metadata components. In other words, it describes what you can do with those components, such as create an object or deploy one. If it helps you to separate them into two parts, you can think of the metadata components as nouns, and the API calls as verbs.

With the Metadata API you can:

- Work with setup configuration as metadata files
- Copy, paste, merge, and manipulate metadata files using familiar tools, such as text editors, IDEs, batch scripts, and source control systems
- Migrate configuration changes between organizations, either between two development environments or from development to production
- Create your own tools for managing organization and application metadata

What is a Force.com Project?

A Force.com project is a collection of metadata files gathered together so that you can work with the files locally. Depending on your needs, you can create a Force.com project that contains a single component or every component in your organization. The latter case, retrieving every portable component from your organization, is not always the best choice because it can make deployment a longer and more difficult process.

Note: The Metadata API has a limit of 1500 files that you can retrieve or deploy at one time. If you need to retrieve or deploy more than 1500 files, you must do so in batches.

A good way to structure your project is to think about what you want to accomplish, and then create a project for only those components. You can later edit the project if you have components you want to add or remove.

Which Metadata Components are Accessible?

The following table lists all of the components that can be retrieved or deployed via the Metadata API, each component's corresponding metadata name, and the folder where the component resides.

Component	XML <name> Metadata Type	Folder
Apex class	`ApexClass`	`classes`
Apex trigger	`ApexTrigger`	`triggers`
Analytic snapshot	`AnalyticSnapshot`	`analyticSnapshots`
Custom application	`CustomApplication`	`applications`
Custom object or standard object	`CustomObject`	`objects`
Custom object translation	`CustomObjectTranslation`	`objectTranslations`
Custom field	`CustomField`	`objects`
Custom label	`CustomLabel`	`labels`
Custom page web link	`CustomPageWebLink`	`weblinks`

Component	XML <name> Metadata Type	Folder
Custom site	`CustomSite`	`sites`
Custom tab	`CustomTab`	`tabs`
Dashboard	`Dashboard`	`dashboards`
Document	`Document`	`document`
Email template	`EmailTemplate`	`email`
Home page component	`HomePageComponent`	`homePageComponents`
Home page layout	`HomePageLayout`	`homePageLayouts`
Page layout	`Layout`	`layouts`
Letterhead	`Letterhead`	`letterhead`
Picklist	`Picklist`	`objects`
Portal	`Portal`	`portals`
Profile	`Profile`	`profiles`
Record type	`RecordType`	`objects`
Report	`Report`	`reports`
Report type	`ReportType`	`reportTypes`
Scontrol	`Scontrol`	`scontrols`
Static resource	`StaticResource`	`staticResources`
Translation workbench	`Translations`	`translations`
Validation rule	`ValidationRule`	`objects`
Visualforce component	`ApexComponent`	`components`
Visualforce page	`ApexPage`	`pages`
Web link	`WebLink`	`objects`
Workflow	`Workflow`	`workflows`

Not everything in a Salesforce organization is available as a metadata component. With each release of Salesforce, more and more metadata components are added to the Metadata API.

The following components cannot be retrieved or deployed in the Metadata API, and changes to them (in a production organization or development environment) require manual migration:

- Account Contact Roles
- Account Partner Roles
- Account Teams
- Activity Button Override
- Activity Settings
- Approval Process
- Auto-number on customizable standard fields
- Business Hours
- Call Center Case Contact Roles

- Campaign Influence
- Case Assignment Rules
- Case Team Roles
- Case Escalation Rules
- Console Layouts
- Contracts Settings
- Delegated Administration
- Dependent picklist rules
- Email Bounce Administration
- Email-to-Case
- Fiscal Year
- Forecasts
- HTML Document and Attachment Settings
- Label renames
- Lead Assignment Rules
- Lead Processes
- Lead Settings
- List views on standard objects
- Ideas Comment Validation Rule
- Ideas Communities
- Ideas Settings
- Mail Merge Templates
- Mobile Administration
- Mobile Users and Devices
- Offline Briefcase Configurations
- Opportunity Big Deal Alert
- Opportunity Competitors
- Opportunity Contact Roles
- Opportunity Sales Processes
- Opportunity Settings
- Opportunity Update Reminders
- Organization Wide Defaults
- Partner Management
- Picklist in customizable standard fields—custom picklist fields on standard objects work in the Metadata API, but not standard picklists
- Predefined Case Teams
- Product Schedule Setup
- Product Settings
- Public and Resource Calendars
- Public Groups
- Queues
- Role Hierarchies
- Salesforce to Salesforce
- Sales Team and Account Team Roles
- Search layouts on standard objects
- Search Settings
- Self-Service Public Knowledge Base
- Self-Service Web-to-Case

- Self-Service Portal Settings
- Self-Service Portal Users
- Self-Service Portal Font and Colors
- Sharing Rules
- Solution Categories
- Solution Settings
- Solution Processes
- Standard picklist fields
- Support Processes
- Support Settings
- Support Auto-Response Rules
- Tab renames
- Tag Settings
- Territory Assignment Rules
- User Interface Settings
- Web Links on PersonAccount page layouts
- Web-to-Lead
- Web-to-Lead Auto-Response

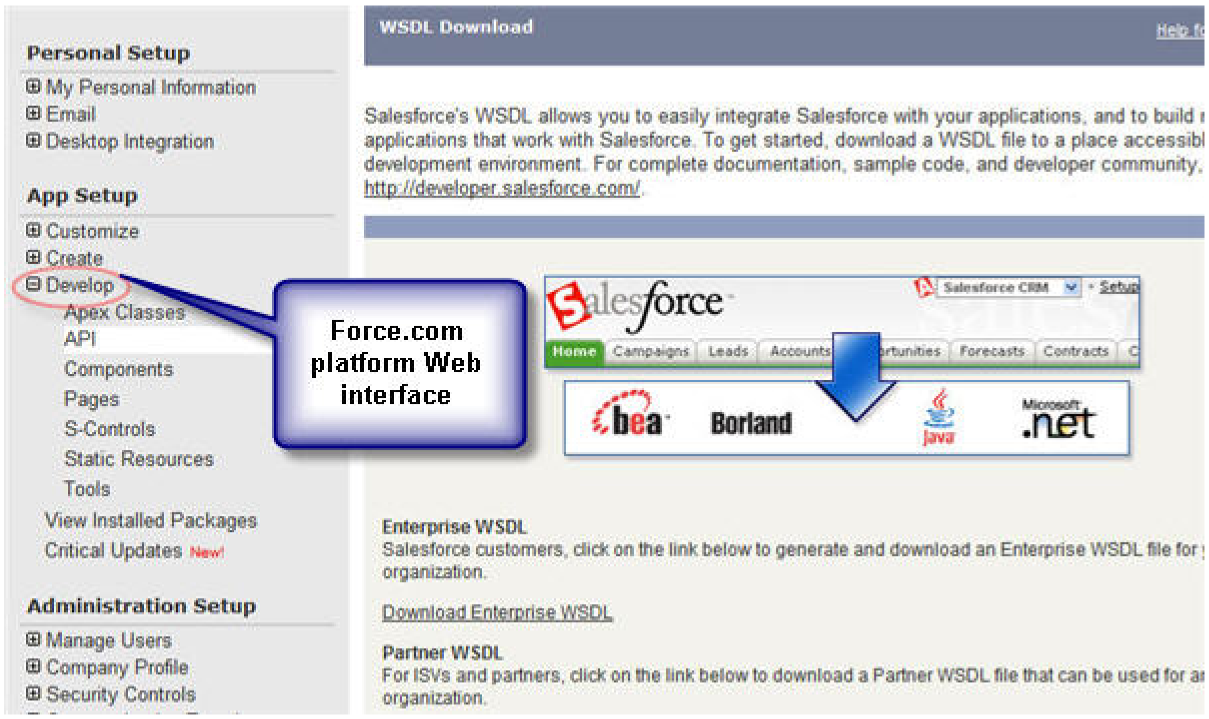
Development Tools

Regardless of your role in the development process, it is important to understand at a high level how all of the Force.com development tools operate, and the development tasks that overlap each other. A challenge of release management is tracking the changes in every organization, and knowledge of the tools makes tracking changes an easier job.

You are probably familiar with how to build applications using the Salesforce Web user interface. Project-based development requires working with desktop tools as well. The Force.com IDE is the best tool for team development, migration of selected components, and writing Apex. The Force.com Migration Tool, on the other hand, is useful for migrating large scale or repetitive changes between environments. These and other Force.com tools are examined in the following sections.

Force.com App Builder Tools

The Force.com Web interface makes application development easy with point-and-click app builder tools. Whenever you click the **Setup** link in a Salesforce organization, you have the ability to modify the metadata of your organization.



Salesforce's WSDL allows you to easily integrate Salesforce with your applications, and to build r applications that work with Salesforce. To get started, download a WSDL file to a place accessibl development environment. For complete documentation, sample code, and developer community, http://developer.salesforce.com/.

Enterprise WSDL
Salesforce customers, click on the link below to generate and download an Enterprise WSDL file for y organization.

Download Enterprise WSDL

Partner WSDL
For ISVs and partners, click on the link below to download a Partner WSDL file that can be used for ar organization.

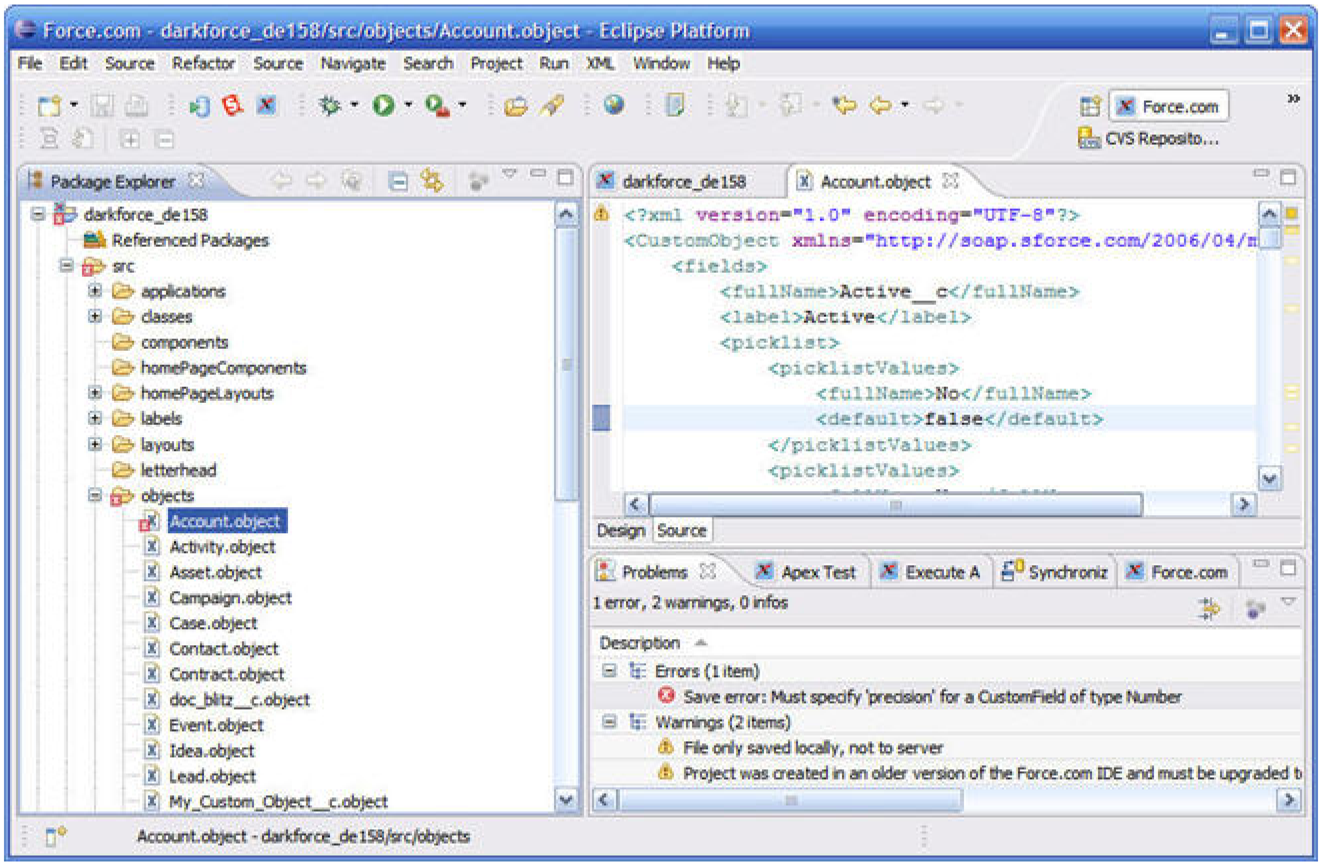
Figure 11: Web-based Declarative App Builder

Because the Web interface is so easy to use, administrators often make changes directly in the production organization, rather than using a development environment. This is great for users who need the functionality immediately, but the change in production needs to be tracked so that you can migrate the same functionality to your testing or staging sandboxes, if necessary.

While there are many tools you can use to create and modify application components, the Web interface is the easiest place to start. For example, you can create a component using the Web interface, and then create many more similar components by using the copy and paste functionality enabled by the Metadata API. In order to leverage the copy and paste functionality (or any other file-based actions), you must first retrieve the metadata to a local project.

Force.com IDE

The Force.com IDE is an integrated development environment for developing applications on the Force.com platform using Apex, Visualforce, and other metadata components. Designed for developers and development teams, the IDE provides tools to accelerate Force.com application development, including source code editors, test execution tools, deployment aids and integrated help.



```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/m
    <fields>
        <fullName>Active__c</fullName>
        <label>Active</label>
        <picklist>
            <picklistValues>
                <fullName>No</fullName>
                <default>false</default>
            </picklistValues>
            <picklistValues>
```

```
1 error, 2 warnings, 0 infos
Description
Errors (1 item)
    Save error: Must specify 'precision' for a CustomField of type Number
Warnings (2 items)
    File only saved locally, not to server
    Project was created in an older version of the Force.com IDE and must be upgraded t
```

Account.object - darkforce_de158/src/objects

Figure 12: Force.com IDE

The Force.com IDE is built on top of the open-source Eclipse Platform, and is available as a plug-in that can be installed into the free Eclipse IDE workbench. Using the Force.com IDE, developers can perform the following tasks:

- Create Force.com projects associated with a Salesforce organization (the home organization) and download metadata component definitions as text files.
- Create and edit Apex, Visualforce, S-controls, and XML metadata components in source code editors that provide syntax highlighting, code assistance, and server-based error checking.
- Test and debug Apex classes and triggers using the Apex Test Runner.
- Run anonymous blocks of Apex on the server in the Execute Anonymous view.
- Browse your schema objects and fields or assemble and execute SOQL queries in the Schema Explorer.
- Synchronize project contents with changes on the server using Save to Server, Refresh from Server, and Synchronize with Server commands.
- Utilize the Compare Editor to merge changes when conflicts are detected.
- Deploy metadata components from one Salesforce organization to another, or validate a planned deployment without saving changes, using the Deploy to Server wizard.
- Manage projects and files using the standard views, tools, and commands in the Eclipse IDE workbench.

Schema Explorer

The Force.com Schema Explorer is accessed through the Force.com IDE by double-clicking the `.schema` file in the Navigation pane.

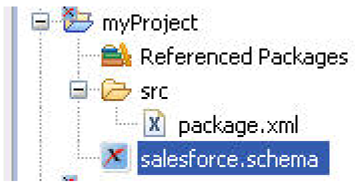
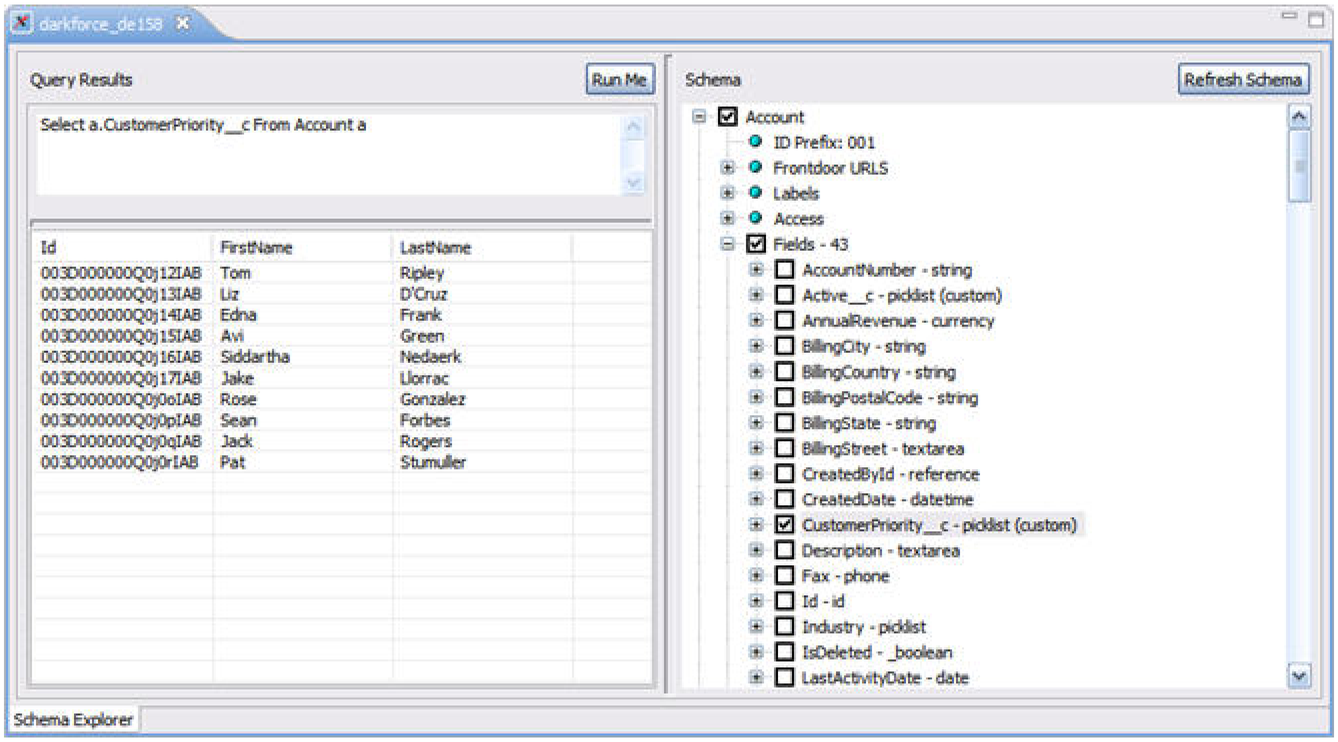


Figure 13: Schema Explorer

The Schema Explorer is a tool for browsing the metadata of a Salesforce organization, and for querying data. The Schema Explorer presents the logged-in user's view of the Salesforce data model, including object visibility, permissions, data types, lookup values and other information that is useful in developing applications on the Force.com platform.

The Schema Browser is depicted as a hierarchical tree view of your organization's schema. Every object that can be accessed by the logged-in user is displayed at the root level of the tree, with additional related properties beneath those roots. Tooltips are also supplied to provide quick summary information without having to drill down into the object.



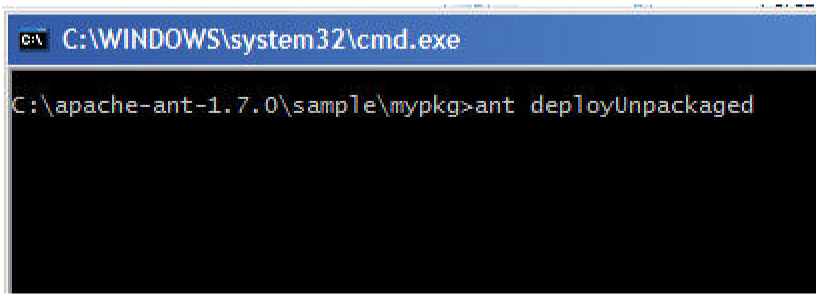
```
Select a.CustomerPriority__c From Account a
```

Id	FirstName	LastName
003D000000Q0j12IAB	Tom	Ripley
003D000000Q0j13IAB	Liz	D'Cruz
003D000000Q0j14IAB	Edna	Frank
003D000000Q0j15IAB	Avi	Green
003D000000Q0j16IAB	Siddartha	Nedaerk
003D000000Q0j17IAB	Jake	Llorrac
003D000000Q0j0oIAB	Rose	Gonzalez
003D000000Q0j0pIAB	Sean	Forbes
003D000000Q0j0qIAB	Jack	Rogers
003D000000Q0j0rIAB	Pat	Stumuller

```
Account
    ID Prefix: 001
    Frontdoor URLS
    Labels
    Access
    Fields - 43
        AccountNumber - string
        Active__c - picklist (custom)
        AnnualRevenue - currency
        BillingCity - string
        BillingCountry - string
        BillingPostalCode - string
        BillingState - string
        BillingStreet - textarea
        CreatedById - reference
        CreatedDate - datetime
        CustomerPriority__c - picklist (custom)
        Description - textarea
        Fax - phone
        Id - id
        Industry - picklist
        IsDeleted - _boolean
        LastActivityDate - date
```

Force.com Migration Tool

The Force.com Migration Tool is a Java/Ant-based command-line utility for moving metadata between a local directory and a Salesforce organization. Unlike the Force.com IDE, the Force.com Migration Tool has no graphical user interface. You choose which components you want to deploy, the server address, and other deployment details by editing control files in a text editor and using command-line arguments.



C:\WINDOWS\system32\cmd.exe

```
C:\apache-ant-1.7.0\sample\mypkg>ant deployUnpackaged
```

Figure 14: Force.com Migration Tool Command-line Interface

Most people will find the graphical user interface of the Force.com IDE easier to use than editing text files and providing arguments on the command line. However, the Force.com Migration Tool is especially useful in the following scenarios:

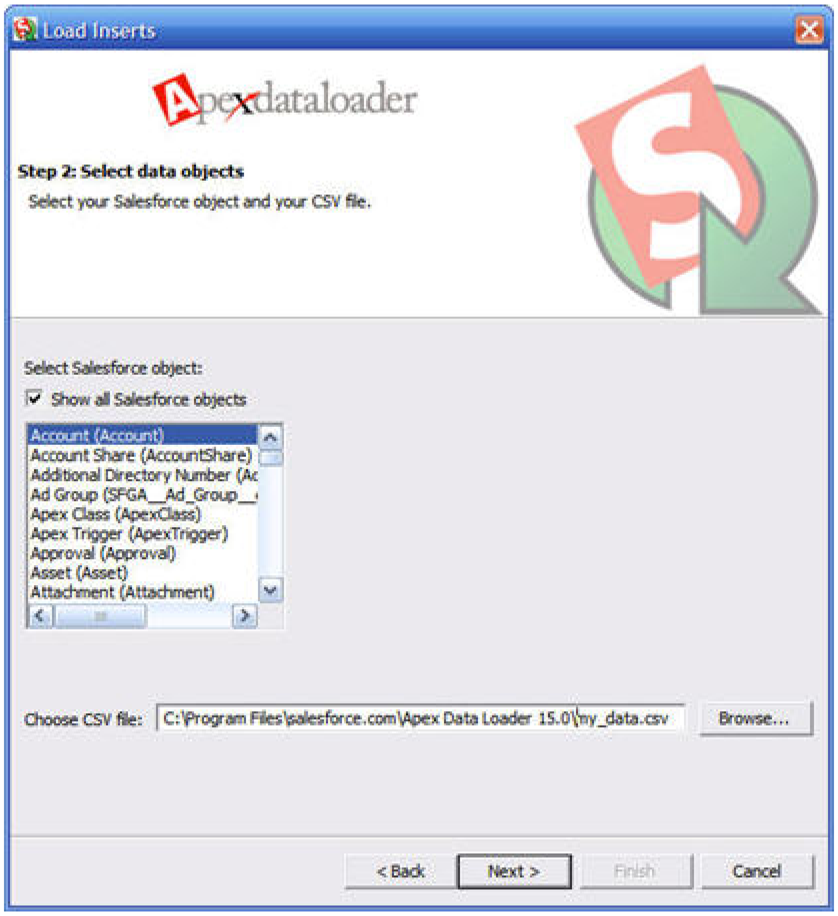
- Development projects where you need to populate a test environment with large amounts of setup changes—making changes with an automated script is far faster than entering them by hand.
- Deployments where you need to change the contents of files between organizations—for example, if you want to change the Running User on a dashboard, you can retrieve the Running User from organization A, make a change, and then deploy the Running User to organization B. If you tried to do this in the Force.com IDE, the IDE would force you to save the change back to organization A (where the organization B user probably does not exist) before launching the Deploy Wizard to deploy to organization B. The Force.com Migration Tool gives you complete control over the `retreive()` and `deploy()` commands; editing and saving a file on your local file system does not change the metadata in any organization until you choose to deploy it.
- Multi-stage release processes—a typical development process requires iterative building, testing, and staging before releasing to a production environment. Scripted retrieval and deployment of components can make this process much more efficient.
- Repetitive deployment using the same parameters—you can retrieve metadata from your organization, make changes, and deploy your changes to an organization. If you need to repeat this process, it is as simple as calling the same deployment target again.
- When migration from staging to production is done by highly technical resources—anyone who prefers deploying in a scripting environment will find the Force.com Migration Tool a familiar process.

Data Loader

The Data Loader is a client application for the bulk import or export of data. Use it to insert, update, delete, or extract Salesforce records.

When importing data, the Data Loader reads, extracts, and loads data from comma separated values (CSV) files. When exporting data, it outputs CSV files.

Note: If commas are not appropriate for your locale, use a tab or other delimiter.



Step 2: Select data objects

Select your Salesforce object and your CSV file.

Select Salesforce object:

☑ Show all Salesforce objects

Account (Account)
Account Share (AccountShare)
Additional Directory Number (Ac
Ad Group (SFGA__Ad_Group__
Apex Class (ApexClass)
Apex Trigger (ApexTrigger)
Approval (Approval)
Asset (Asset)
Attachment (Attachment)

Choose CSV file: C:\Program Files\salesforce.com\Apex Data Loader 15.0\my_data.csv

Figure 15: Data Loader

The Data Loader can be useful for loading data into testing environments, such as configuration-only sandboxes, which do not contain any data. For regression testing, it is advisable to use a stable set of data that does not change from release to release, so it is useful to store a set of data locally and use the Data Loader to populate your test environment. The Data Loader has a command-line interface so that scripts can automatically load data into a fresh organization. For more information about the Data Loader, see the Data Loader Developer Guide.

Note: The Data Loader is strictly a tool for loading data, not metadata. You cannot add or delete fields on an object, or load any kind setup changes using the Data Loader.

Tracking Development Changes

One challenge of application lifecycle development is tracking changes. It sounds simple in concept, but in practice, it may not be. Changes can occur simultaneously in both production and development organizations, and in components that are, and are not, available in the Metadata API.

The easiest way to track changes is to use traditional development tools to diff, merge, and version the code. However, these tools only work with components available in the Metadata API, and so it is necessary to manually track changes. This may seem like a lot of work, but being able to track and replicate changes in all environments is the only way to ensure that functionality can be successfully deployed without overwriting changes on the production organization. On the Force.com platform, the most important reason to track changes is because you may have to manually migrate some changes from one organization to another.

Tracking Changes Manually

Any changes you make to components that are available in the Metadata API can be tracked and merged using desktop tools. If you use the Force.com IDE or Force.com Migration Tool, you can put your files in a version control system, and you can track changes using the version control system's built-in functionality. However, most IT organizations make numerous changes to components that are not represented in the Metadata API, and thus require manually tracking changes for both production and development organizations.

It is important to have a method for tracking all changes, and especially important if those changes require manual migration. A useful tool is a change request form that users and administrators fill out for every enhancement requested or change performed. You can create a custom application within Salesforce to record change requests and the actual changes made.

Tip: The AppExchange includes custom apps that you can install to manage this process, (such as the free Change Control app).

Most IT organizations also use a spreadsheet to log and track those changes. Spreadsheets allow you to sort by headers, create pivot tables, and perform other useful operations when migrating or tracking those changes. On the spreadsheet list every change that occurred, including:

- Who made the change
- The organization where the change occurred
- Date and time
- Which component was changed

When do you need to track changes manually?

- Changes made to components not in the Metadata API—You must manually track every change to components that are not available in the Metadata API.
- Changes made using the Salesforce Web user interface—Even if the components are available through the Metadata API, you should track changes made using the Web tools. Concurrent changes between the Web tools and the Metadata API often create dependencies and are a common source of deployment problems. To be on the safe side, it is better to manually track all changes made through the Web interface.

Changes made in the Salesforce Web user interface are logged in the setup audit trail. Using the audit trail in conjunction with your manual change tracking tool is the best way to ensure you do not miss any changes. How you manage this process is up to you, but it is a good idea to regularly transfer all changes in the audit trail to your change list (once a week, for example), or use the audit trail only to cross-check changes on your change list. For more information on using the setup audit trail, see "Monitoring Setup Changes" in the Salesforce online help.

Tracking Changes Using the Force.com Migration Tool

The Force.com Migration Tool is especially useful for script-based deployments, such as migrating batches of components. Another particularly useful batch script can be written to diff metadata changes. Such a script would allow you to perform diffs at scheduled times, or with different sets of components.

1. Create a `package.xml` file that lists the components you want to compare.
2. Create a retrieve target to download the components. For example:

```
<target name="retrieve-production">
   <sf:retrieve
      username="${sf.username}"
      password="${sf.password}"
      serverurl="${sf.serverurl}"
      retrieveTarget="production"
      unpackaged="package.xml" />
```

3. Write an external script to diff the files.
4. Specify a target that calls the script and outputs the results to a file. For example:

```
<target name="show-production-differences">
   <!-- get metadata from organization -->
   <antcall target="retrieve-production"/>

   <!-- perl script to find changes -->
   <exec executable="/bin/bash">
      <arg value-"-c" />
      <arg value-"cd production;
         svn stat|../showdiffs.perl>../report.txt"/>
      </exec>
</target>
```

Note: This procedure is only an example of the basic steps you would take to perform a programmatic diff. You might want to define more than one folder and `package.xml` file so that you can retrieve components in batches.

Tracking Changes Using the Force.com IDE

Any change made in the Force.com IDE can be easily tracked and merged using the built-in features within Eclipse, or any other change-tracking utility. User assistance for Eclipse is beyond the scope of this guide, for more information, click Help in the Force.com IDE.

Synchronizing Changes

The Metadata API describes Force.com components as text-based files that can be transported between different environments. Using third-party tools, these files can be stored, versioned, and divided between developers and across development environments. Team development allows you to do more, but certainly not with less. The complexity of your application development lifecycle increases as you introduce such concepts as multiple development environments, code branches, versions, etc.

Synchronizing all those changes can be a challenge. For example, you might have one development sandbox dedicated to enhancements to existing applications on your production organization, and another development sandbox for new applications. Each of those organizations could have multiple projects, all making changes at the same time. Merging those changes requires two separate processes: synchronizing changes between projects in the same organization, and then integrating the changes between organizations so that you know everything works when combined.

Synchronizing Changes

A fundamental difference between traditional software development and cloud computing is that in cloud computing the server always has the true definition of the components. The files you work with in a local Force.com project are a representation of the objects on the server. This is an important distinction because synchronization does not happen between projects directly, but between each project and the server:

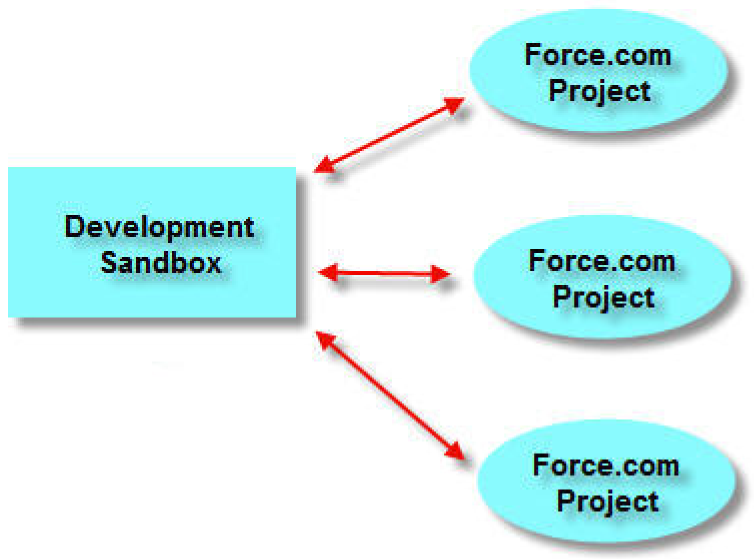


Figure 16: Multiple Projects Connect to a Single Organization

If you use the Force.com IDE, synchronizing your changes with the home organization is as easy as clicking the **Save** button. For changes that occur simultaneously in both organizations, the synchronization tools allow you to overwrite files in either direction, or merge the changes. How to save, synchronize, and refresh files is covered in the Force.com IDE help.

Integrating Changes Across Development Environments

If you have more than one development environment, you must merge all changes into one organization from which you can deploy. If you have only two sandboxes, you can migrate changes back and forth between them fairly easily. Each sandbox can be used to deploy changes to the other organization, or to production.

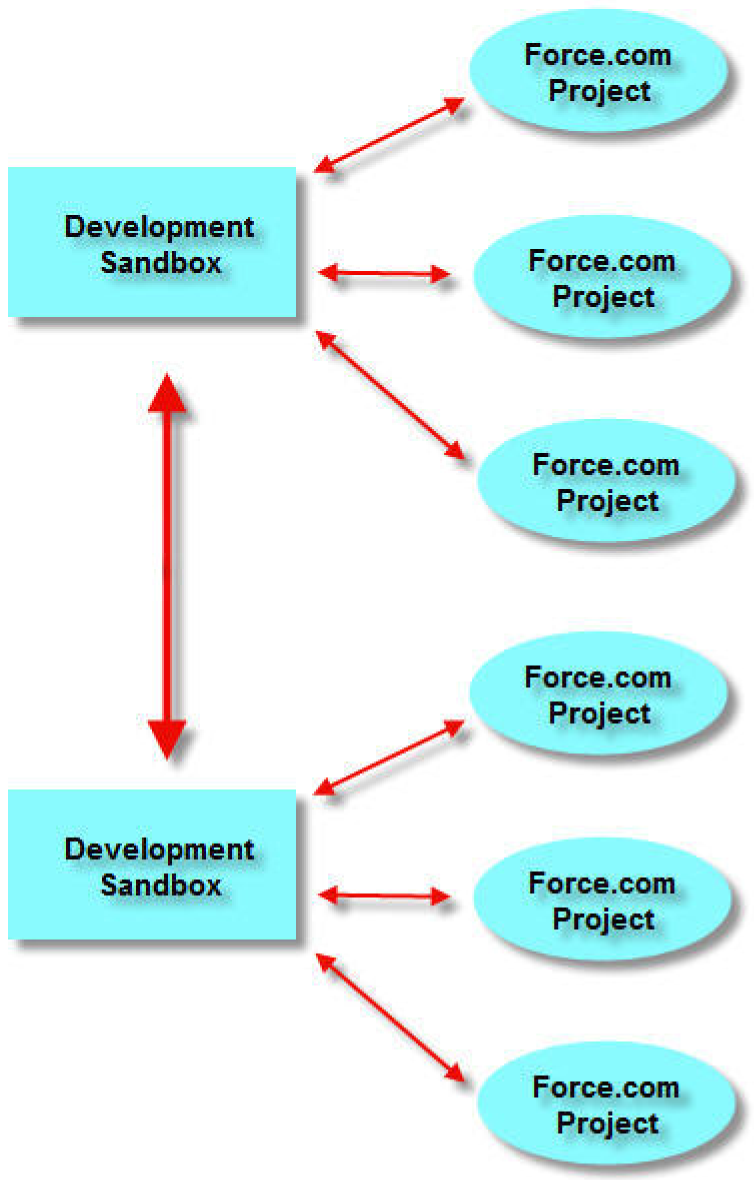


Figure 17: Synchronizing Two Development Sandboxes

As you add development environments, the complexity of synchronizing them goes up exponentially. In this case, it is much easier to merge the changes in a separate integration sandbox organization than with each other. In such a scenario, migration from development sandbox to integration is one way only:

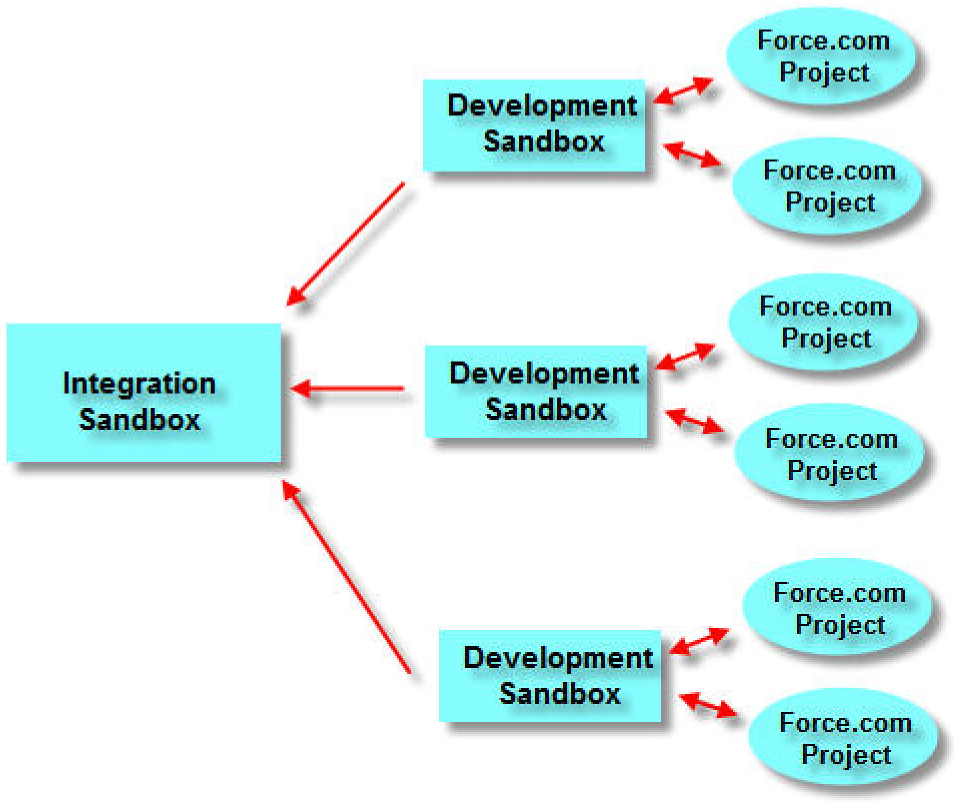


Figure 18: Integrating Multiple Development Sandboxes

Chapter 4

Migrating Changes between Environments

Migration is the act of moving configuration changes from one Salesforce organization to another. During the development cycle, you might migrate many times, either to keep development organizations in sync or to move changes through development organizations toward production.

Migration can happen in two ways: manually or through the Metadata API.

- Metadata migration—Components that are available in the Metadata API can be migrated using desktop tools.
- Manual migration—Changes to components that are not available in the Metadata API must be manually migrated in each environment. That is, you must repeat the exact same modifications in every production or development organization. Manual migration is made easier by diligently tracking changes.

The typical steps to migrate changes are:

1. Determine which components need to be migrated first, This is especially important if you have both manual and metadata migrations to perform. For example, if you need to create a queue against a custom object, the custom object must be migrated first.
2. Migrate components in the order required:
   - Look at your change list to determine if anything needs to be manually migrated. If so, log into the organization you will be migrating changes into and make those setup changes manually.
   - Retrieve the latest metadata changes from the server.
3. Optionally modify your Force.com project to deploy only a subset of components.
4. Deploy.

Migrating Changes Manually

Manual migration requires performing setup changes through the Salesforce Web user interface. For example, suppose you require a new approval process on your production organization, but you want to develop and test it in a sandbox before you expose it to users. Because approval processes are not available in the Metadata API, the only way to migrate the approval process to your production organization is to manually repeat the setup steps you did when you created the approval process in sandbox.

You might think that the easiest way to avoid manual migration is to make all of your changes on production and then refresh your sandbox. While this is possible, sandboxes can only be refreshed every 29 days, and there are other important considerations. Also, any changes on production must be manually migrated in each of your development organizations. Because component dependencies play a large role in deployment, changes you make on your production organization may prevent you from deploying applications you develop on sandbox. It follows that if you have multiple development organizations, you will have to manually migrate changes from production to sandbox many times. For these reasons, developing on production is potentially much more difficult than manually migrating changes from sandbox to production.

Note: Many customizations and features can be developed on production, but these changes should require no testing and little training.

The best way to manage manual migrations is to establish a change process on your production organization, and to track the changes that require manual migration.

How Files are Migrated

Migrating changes from one organization to another using metadata files requires an intermediate tool that interacts with both environments via the Metadata API. The following figure shows how changes are migrated from sandbox to production.

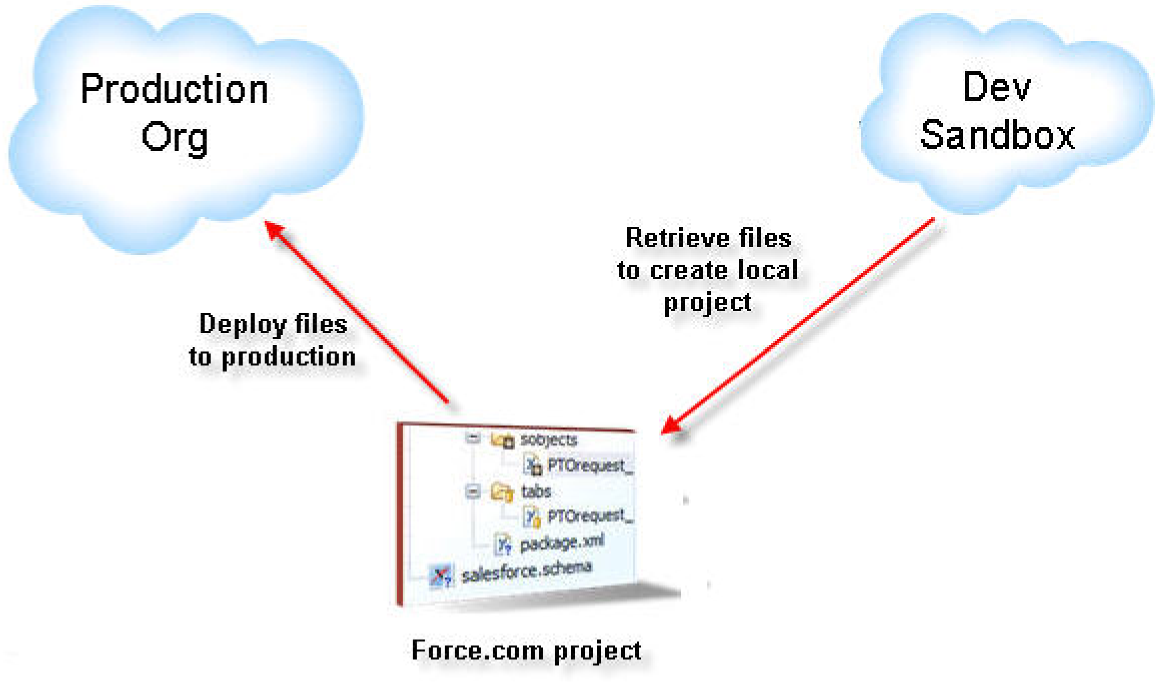


Figure 19: Migration Between Organizations

You might be wondering why you need a local project to migrate files that are stored in the cloud. This is because the Metadata API was designed to support traditional software development tools that operate on source files, such as text editors, diff/merge utilities, and version control systems, all of which require a local file system. Once you create a Force.com project, you can deploy directly to the production organization any number of times; you do not need to retrieve files unless you want to synchronize with the server.

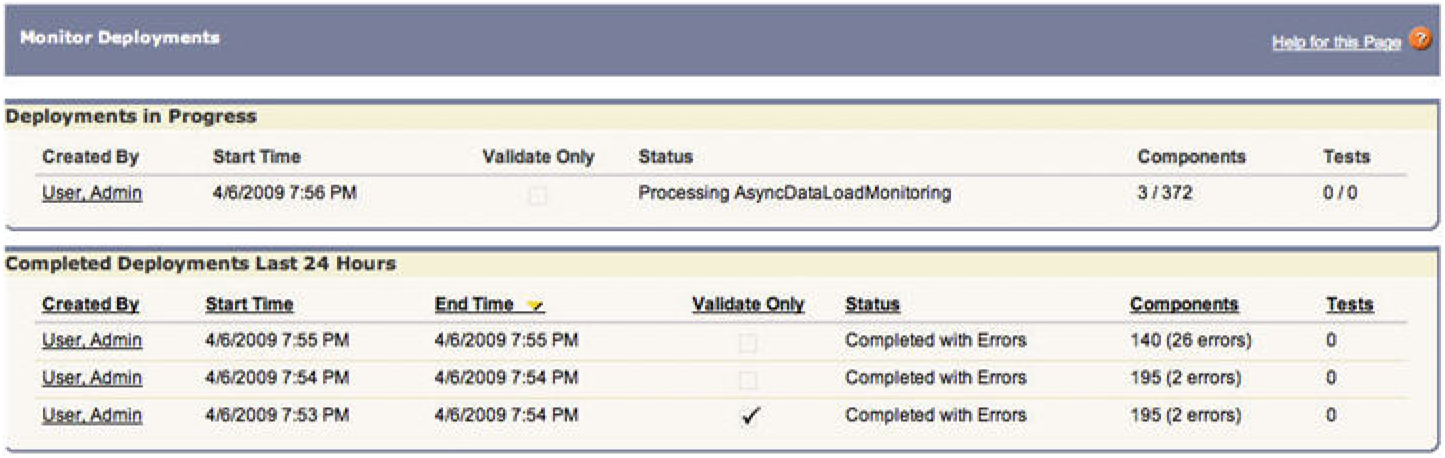
What Affects Deployment Time?

Migrating metadata from a local directory to a Salesforce organization is accomplished by calling the Metadata API `deploy()` method. Both the Force.com IDE and the Force.com Migration Tool use this method, and so there should be very little difference in deployment time when using either tool. The `deploy()` method is asynchronous, meaning that it may not return results immediately, and there are several factors that determine how quickly those results are returned:

- Number and size of files—The more you have to deploy, the longer deployment takes. However, network payloads are rarely larger than 10 MB, so raw file size usually does not play a significant role.
- Type of components—Some components take longer to process than others. For example, custom fields, custom junction objects, and profiles take longer to deploy than other components.
- Processing time—Making a change that requires recalculating data takes an amount of time proportional to the amount of data that has to be modified. For example, if you were to change a field type, this could require modifying all of the records that use that field.
- Test execution—When deploying to a production organization, the number and complexity of Apex tests has a large impact on the deployment time.
- Network and server availability—These are of minimal concern compared to other factors. However, you may want to schedule long-duration deployments during off-peak hours so that you are not waiting on deployments during working hours, or locking components from use.
- Locking—If users are working in the organization during deployment, this can interfere with both users and the deployment. Users may be locked out of components, or the deployment may be waiting for a user to release a lock. The longest running processes are those that reorganize users (or groups of users) who own large numbers of records. For example, changing a sharing rule does not take much time if the rule shares 100 records owned by three users with another six users, but moving one user on a role or territory hierarchy takes a long time if that user owns a gigabyte of records. Other ways users can force the processing of large numbers of records include the following:
  - Creating portal users or users affected by forecasting
  - Creating, modifying, or deleting sharing rules
  - Moving or changing users, roles, groups, account owners, or team membership, when there are a large number of associated records

Monitoring Deployment

The easiest way to find out the status of a deployment is to look at the Monitoring Deployments page. This page lists various statistics about the deployment, such as who initiated it, the duration, status, number of components, and number of tests run. The Monitoring Deployments page tracks deployments that are in progress and those completed in the last 24 hours.



Deployments in Progress

Created By	Start Time	Validate Only	Status	Components	Tests
User, Admin	4/6/2009 7:56 PM	☐	Processing AsyncDataLoadMonitoring	3 / 372	0 / 0

Completed Deployments Last 24 Hours

Created By	Start Time	End Time	Validate Only	Status	Components	Tests
User, Admin	4/6/2009 7:55 PM	4/6/2009 7:55 PM	☐	Completed with Errors	140 (26 errors)	0
User, Admin	4/6/2009 7:54 PM	4/6/2009 7:54 PM	☐	Completed with Errors	195 (2 errors)	0
User, Admin	4/6/2009 7:53 PM	4/6/2009 7:54 PM	✓	Completed with Errors	195 (2 errors)	0

Figure 20: Monitor Deployments Page

To view the Monitoring Deployments page, click **Setup** ➤ **Deploy** ➤ **Monitor Deployments** in Salesforce.

Note: For troubleshooting deployments that are taking longer than anticipated, look at the `Status` field to see if the deployment is spending a disproportionate amount of time on a particular metadata type (for example, Profile), or a specific component (for example, Mileage__c), or running Apex tests. Keeping track of the duration of various parts of the deployment process can help you troubleshoot precisely where deployment is taking a long time. Monitoring your deployment closely can also help you decide which components you should group together if you need to process your deployments in batches.

For more information on the Monitoring Deployments page, see "Monitoring Metadata Deployments" in the Salesforce online help.

Deployment Dependencies

Whether you are migrating changes between development environments or deploying changes to a production organization, there are a number of factors that affect whether or not the deployment will succeed. The most significant of these factors are *dependencies*.

There are a few different kinds of dependencies:

- **Parent-child**—Metadata components may depend on other components. For example, you cannot deploy a custom field without deploying its custom object as well because the field's existence depends on the object. These kinds of object dependencies are not always within the same object; they can be across different objects. For example, a relationship field on one object cannot be deployed without the target object being included in the deployment, or already present in the target organization.
- **Referenced file**—Every file that is referenced by another file must be either included in the deployment plan, or already in the destination organization. For example, if you have a Visualforce page that references an image as a static resource, if the image does not exist in the target organization, you must include the image in the deployment. Note that the referenced file must be available through the Metadata API. For example, if that same Visualforce page had referenced an image in a personal document folder, and you included all folders in the deployment plan, the deployment would fail because personal documents are not available via the Metadata API.
- **Ordering**—Dependencies require that components are deployed in a specific order, and within a single deploy operation, this order is handled automatically by the Metadata API. However, if you deploy a subset of components, or split your deployment into multiple batches, you must take into account the ordering dependencies yourself.
- **Mandatory fields**—When you create an object using the Force.com IDE or the Salesforce Web user interface, the tool enforces the creation of mandatory fields. However, when working with files in a local directory, it is possible to create or modify metadata files that are missing mandatory fields and therefore cannot be deployed. You can also introduce these kinds of errors by copying and pasting between components.

Migrating Files in Batches

The easiest way to migrate changes is to deploy all of them in a single operation. However, there are a number of reasons you might want to divide the deployment into smaller pieces:

- The deployment contains over 1500 files—The maximum number of files that can be deployed (or retrieved) at one time through the Metadata API is 1500. If you have more than 1500 files, you must deploy components in batches.
- Long deployments—If you experience unusually long deployments, you can divide your deployment into smaller pieces. This can reduce user impact due to locks being held in long-running operations.
- Apex testing—You might want to divide your components into two parts: those that require testing and those that do not. This speeds the deployment of components that do not require testing and locks fewer components.

Determining Which Files to Deploy Together

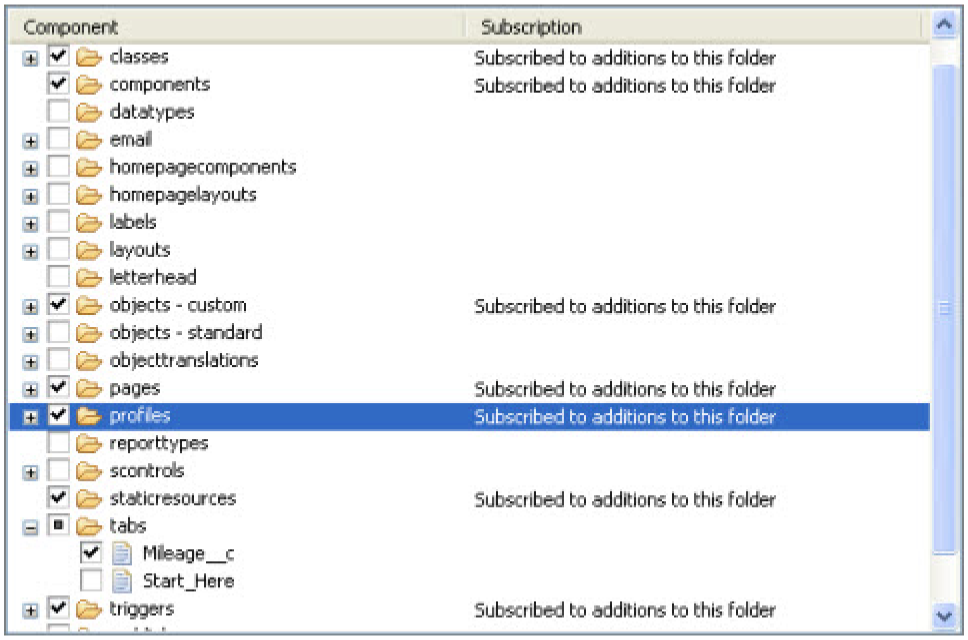
If you need to divide your deployment into smaller pieces, it is important to know which components to deploy at the same time:

- Deploy components that do not trigger tests—Some components cannot cause a test to break, and so they do not trigger tests on a production organization. If the deployment consists entirely of the following components, you can deploy to production without running tests:
  - Visualforce component (ApexComponent)
  - Visualforce page (ApexPage)
  - Dashboard
  - Email template
  - Report
  - Scontrol
  - Static resource
- Do not split dependent components—Because file dependencies play such a large role in migrating metadata, it is very important not to separate dependent components. If you do not know which components are dependent, then migrate the most numerous components separately.
- Deploy the most numerous components separately—The following components can be the most numerous, and so you may want to deploy these separately from other components.
  - Email templates can be deployed separately, but must be deployed before the components that use the templates.
  - Dashboards can be deployed separately, but should be deployed before reports in case a custom button links to a report.
  - Reports may contain the largest number of components, and they can be deployed after all other components and in multiple batches.

Deploying Batches of Files Using the Force.com IDE

There are two ways you can use the Force.com IDE to deploy batches of components: either by changing the project contents or deselecting the components you do not want to deploy when you use the Force.com Deployment Wizard.

The easiest way to deploy batches of files is to create a new Force.com project and use the Choose Metadata Components dialog to select only the components you want to deploy. The Choose Metadata Components dialog is a graphical tool that allows you to select individual components or all components of a certain type. When you create a project, you use this dialog to determine which components you want in the project, but you can edit the project contents at any time, as well.



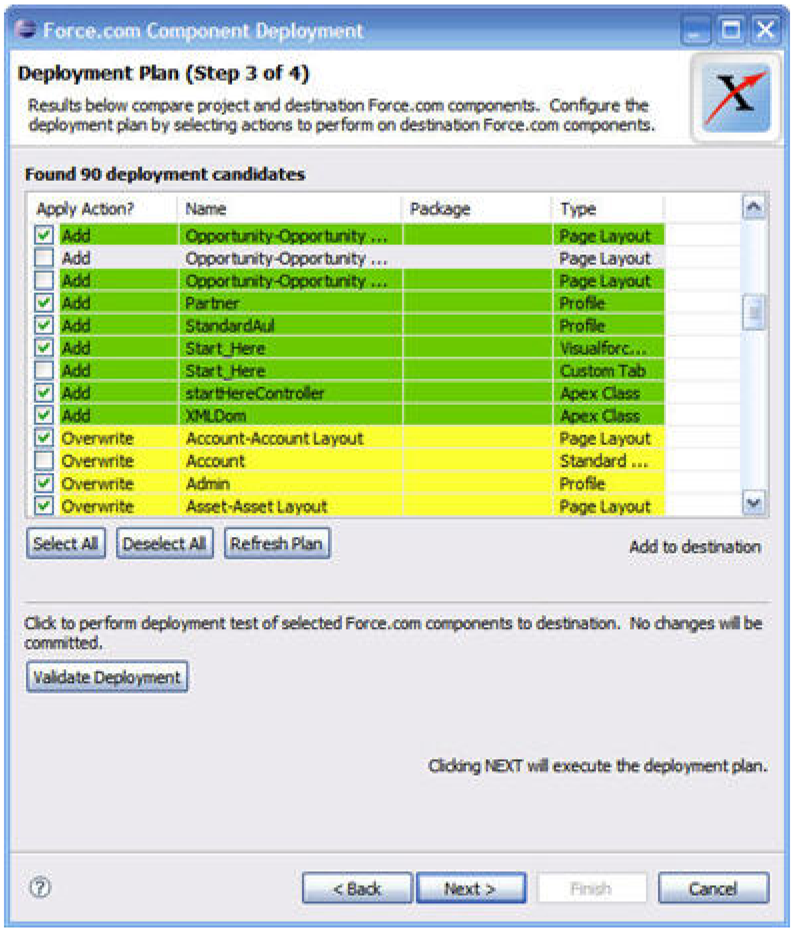
Component	Subscription
✓ classes	Subscribed to additions to this folder
✓ components	Subscribed to additions to this folder
datatypes	
email	
homepagecomponents	
homepagelayouts	
labels	
layouts	
letterhead	
✓ objects - custom	Subscribed to additions to this folder
objects - standard	
objecttranslations	
✓ pages	Subscribed to additions to this folder
✓ profiles	Subscribed to additions to this folder
reporttypes	
scontrols	
✓ staticresources	Subscribed to additions to this folder
■ tabs	
✓ Mileage__c	
Start_Here	
✓ triggers	Subscribed to additions to this folder

Figure 21: Choosing Metadata Components in a Project

When you create or edit your project contents in this manner, what happens behind the scenes are modifications to the `package.xml` file. This file is also called the *project manifest*, and determines what is in your project, both when you retrieve and deploy components. You can also edit the `package.xml` file by hand by dragging it into the Editor pane.

Caution: Editing the `package.xml` file by hand is only advisable if you must have finer control than the dialog can give you. Note that if you open the Choose Metadata Components dialog after editing the `package.xml` file, you may undo some of the changes you made.

The other way to control the deployment plan is to use the Force.com Deployment Wizard. This wizard allows you to choose which components in your project you want to deploy, and the specific action you want to take: add, delete, overwrite, or take no action. To use the wizard, right-click a Force.com project and choose **Force.com** ➤ **Deploy to Server**.



Force.com Component Deployment

Deployment Plan (Step 3 of 4)

Results below compare project and destination Force.com components. Configure the deployment plan by selecting actions to perform on destination Force.com components.

Found 90 deployment candidates

Apply Action?	Name	Package	Type
✓ Add	Opportunity-Opportunity ...		Page Layout
☐ Add	Opportunity-Opportunity ...		Page Layout
☐ Add	Opportunity-Opportunity ...		Page Layout
✓ Add	Partner		Profile
✓ Add	StandardAul		Profile
✓ Add	Start_Here		Visualforc...
☐ Add	Start_Here		Custom Tab
✓ Add	startHereController		Apex Class
✓ Add	XMLDom		Apex Class
✓ Overwrite	Account-Account Layout		Page Layout
☐ Overwrite	Account		Standard ...
✓ Overwrite	Admin		Profile
✓ Overwrite	Asset-Asset Layout		Page Layout

Add to destination

Click to perform deployment test of selected Force.com components to destination. No changes will be committed.

Clicking NEXT will execute the deployment plan.

Figure 22: Choosing Components in the Deployment Wizard

How to Migrate Batches of Files Using the Force.com Migration Tool

If you use the Force.com Migration Tool, there is no graphical user interface to edit the `package.xml` file, and so you must become familiar with the format of the XML files and edit this file by hand. There are two deployment targets you can call that will help you determine which components to include in each `package.xml` file you must create.

- `describeMetadata`—Returns a list of metadata types that are enabled for your organization. This target is useful when you want to identify the syntax needed for a metadata type in `package.xml`.
- `listMetadata`—Returns the names and other properties of individual metadata components in your organization, along with extra information about each one. This target is useful when you want to include specific components in the `package.xml`, or if you want a high-level view of particular metadata types in your organization. For example, you could use this target to return a list of names of all the reports in your organization, and use this information to create a `package.xml` file that returns the specific reports you want to migrate.

For more information about using these deployment targets, see the Force.com Migration Tool Guide.

Deleting and Renaming Components

When you migrate components from one organization to another, the operation is similar to an upsert. That is, you can deploy changes to a component that already exists, but if the component does not exist, it is created. For example, if you have a component named `foo` in organization A and you change a data type, when you deploy that change to organization B, the data type is changed as long as `foo` exists in organization B. However, if `foo` does not exist in organization B, the entire component is created.

If you rename a component in organization A and deploy that component to organization B, you might expect the name to be changed in organization B, but instead, you create a new component in organization B. This is because the deploy operation looks for matching names—since a component with the name does not exist in organization B, a new one is created. For example, if you rename `foo` to `foos` in organization A, and then deploy that change to organization B, this results in two components, both `foo` and `foos`, in organization B.

To rename a component, you must delete the component, and then recreate it with a new name. The process you use to delete a component depends on the environment:

- For development and testing environments—Delete the components, recreate them with new names, and reload test data.
- For production or staging environments—Rename components using the Salesforce Web user interface. This preserves the data in existing records.

The project manifest (`package.xml`) determines what is deployed in a project, but this file cannot be used to process deletes. In order to delete files in a Force.com project, you must create a project manifest and name it `DestructiveChanges.xml`. When you include this file in a deployment, the components you specify for deletion are removed in the target organization.

Using the AppExchange to Migrate Changes

It is possible to use the AppExchange to move metadata between organizations, but it is not an efficient way to migrate changes. Unmanaged packages do not allow you to install components of the same name, and so those components cannot be further modified (via an unmanaged package) after the initial installation.

Another kind of package is a *managed package*. Managed packages are used for distributing versioned applications and add constraints that make them poorly suited to use as an IT development tool. Furthermore, while any type of organization can be used to create an unmanaged package, managed packages must be created using a Developer Edition organization. Managed packages are beyond the scope of this document. For more information, see the topics listed in Additional Resources on page 6.

Note: Unmanaged packages can be a useful for distributing initial components to multiple organizations. For example, if you want all of your development environments to have the same set of core Apex classes, you could package these and distribute them on the AppExchange. This use of unmanaged packages is a convenient way to deliver files to multiple development environments, but cannot be used to make further changes to those files.

Chapter 5

Testing

Testing encompasses everything from verifying that the smallest piece of code works to determining if the finished application meets the original requirements. Because testing means different things at various points in the application development lifecycle, it is useful to look at the different kinds of tests and when they occur:

- Unit tests—A unit is the smallest testable part of an application, usually a method. A unit test operates on that piece of code to make sure it works correctly. Unit tests are usually written during development, in the organization where development takes place.
- Integration tests—The purpose of integration testing is to find inconsistencies between development projects. Projects can be developed in multiple environments, and eventually need to integrate. Force.com integration tests usually occur in a separate organization so that separate projects and environments can be combined and tested in one organization.
- System tests—A type of *black box* testing where the tester does not need any knowledge of the inner workings of the application or its source code. System tests evaluate the behavior and limits of the application beyond what might be expected from real-world data volumes and batch loads. Load testing is often a part of system testing, but this is not necessary on the Force.com platform. Salesforce is regularly load tested, and additional load testing on your part would be redundant and a misuse of server time.
- User acceptance tests—A process used to make sure that the application meets the planned requirements. These tests are usually performed by end users, and gauge how the system will perform in production. The purpose of user acceptance tests is not to uncover bugs, but to gauge satisfaction.
- Staging tests—Before an application is deployed to production, you can test the deployment process by deploying to a staging environment first. A staging environment is used specifically to test deployment, so it should resemble the production environment as closely as possible.
- Regression tests—Regressions usually occur as an accidental consequence of software updates, causing existing functionality to stop working. Common methods of regression testing include running previously run tests and checking whether bug fixes have re-emerged. An important component of regression testing is a stable set of test data that does not change from release to release.

Preparing a Test Environment

The easiest way to prepare a dedicated testing environment is to create a full-copy sandbox of your production organization. However, full-copy sandboxes are expensive, not always necessary, and you may want to load your own data. For these reasons, it is often better to create a configuration-only sandbox and deploy directly from your development sandbox to your testing sandbox. In this way, you will test not only the finished application, but the procedure used to deploy the completed application. The kind of sandbox and amount of data you will load depends on the kind of testing you want to do. For more information see About Sandbox Organizations on page 18.

If you do not have a sandbox, you can use a Developer Edition organization as your testing environment. There are some limitations with this approach. First, Developer Edition organizations have limited licenses, so you cannot have many testers at the same time. Also, this method requires an extra step, which is to populate the Developer Edition organization with a production-like environment, including both metadata and data.

Once you have created your test environment, you can load data into it. You may also have to configure the environment so that users have the appropriate permissions, and that external endpoints, URLs, and other environmental dependencies are adjusted.

Copying Data to Your Organization

Testing often requires loading data into your organization so that users or automated tools can run tests. If you are using a full-copy sandbox, your production data will already be in place so you may not need to load data at all. Otherwise, you will want to export data as a CSV (comma-separated value) file from your production organization, and load it in your development environment using a data import tool.

Note: Some kinds of tests require using a static set of data that is loaded each time a testing organization is created. Static data is especially important for *regression tests*, which determine why something that worked in a previous version no longer works. Because production data changes over time, a full-copy sandbox is not always the best organization to test against.

To import data, you can use either the import wizards or the Data Loader. The Data Loader complements the Web-based import wizards that are accessible from the Setup menu in the online application. Refer to the following guidelines to determine which method of importing best suits your business needs:

Use Web-based importing when:

- You are loading less than 50,000 records.
- The object you need to import is supported by the web-based import wizards.
- You want to prevent duplicates by uploading records according to account name and site, contact email address, or lead email address.

Use the Data Loader when:

- You need to load 50,000 or more records.
- You need to load into an object that is not yet supported by web-based importing.
- You want to schedule regular data loads, such as nightly imports.
- You want to be able to save multiple mapping files for later use.
- You want to export your data for backup purposes.

For more information on importing data, see "Importing Overview" in the Salesforce online help and the Data Loader Developer Guide.

Unit Testing

The Apex language includes the ability to define and execute *unit tests*, which are pieces of code that verify your application works as intended. Each unit test is defined as a test method, which you can execute to see which tests are passing or failing. Unit tests give you instant insight into the quality of your code and provide an early warning system for detecting regressions as you make changes.

There are two common ways to measure code quality using unit tests: *code coverage* and *test case coverage*. Code coverage identifies which lines of code are exercised by a set of unit tests, and which are not, and it is reported as a percentage. This helps you identify sections of code that are completely untested and therefore at greatest risk of containing a bug or introducing a regression in the future. Each time you run a set of unit tests on the Force.com platform, a code coverage number is returned, along with a list of uncovered lines of code in the classes and triggers invoked by the tests.

Note: The Force.com platform requires at least 75% of your code to be covered by automated tests before you can deploy it to a production organization. Ideally, you should strive for 100% coverage. The code coverage restriction is not enforced for sandbox or Developer Edition organizations.

The second method of measuring code quality with unit tests is known as test case coverage. Test cases are the expected real-world scenarios in which your code will be used. It is possible to have 100% code coverage but still have bugs in your code, if the test values you are using do not reflect the full set of real-world possibilities, including corner cases. Test cases are not actual unit tests, but are documents that specify what your unit tests should do. High test case coverage means that most or all of the real-world scenarios you have identified are implemented as unit tests.

When and how you test is up to you, but the longer you wait, the more expensive testing becomes. As a best practice, you should strive for 75% code coverage during development, and not put this off for later testing. Even though development environments do not automatically run tests, it is a good idea to manually run tests every time you migrate changes so you are prepared when it is time to deploy to production.

Fixing Bugs

Where you make bug fixes depends on the severity of the bug and on the specifics of your IT department. Generally speaking, there are two choices: make the fix in the production organization or in a development environment. The best practice is to make all your changes in one place, and then follow a repeatable process for moving the change to production. One version of this process is shown in the following image:

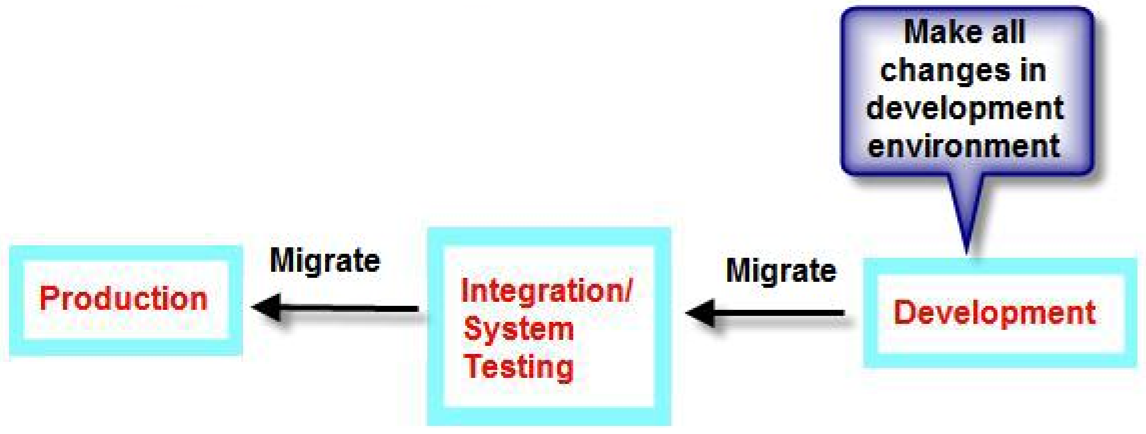

Make all changes in development environment
Production
Migrate
Integration/ System Testing
Migrate
Development

Figure 23: Make Changes in Development and Migrate to Production

This process is not always practical when you have high-priority bugs fixes that need to get to production immediately. In that case, you must fix the bug in the production environment and then migrate the same bug fix to your development environments. This is important because changes made to the production organization can be overwritten when you migrate changes from a development environment. One version of this process is shown in the following image:

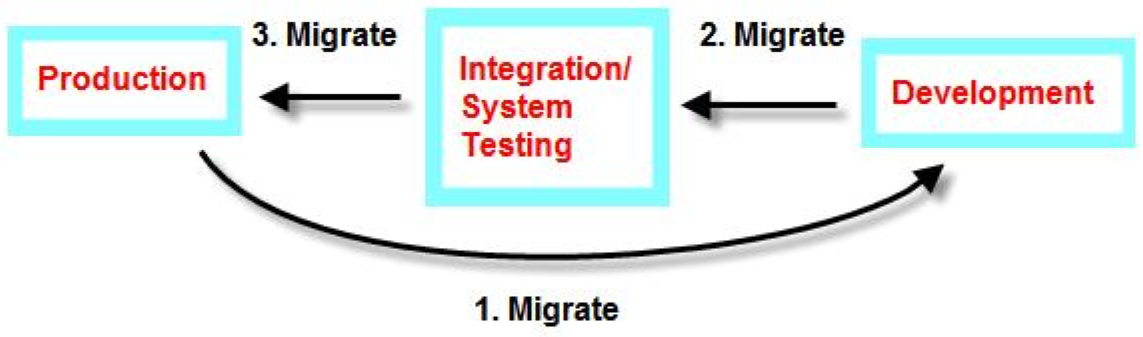


Figure 24: Production Change Migrated to all Environments

Chapter 6

Deploying to Production

The tools and processes for deploying to a production organization are similar to those for migrating changes from one development environment to another. However, when deploying to production, there are some important differences and several additional steps. Because the procedure you take when deploying to your production organization depends on your IT department's policies and on what you are deploying, there is no prescribed process for deploying to production. However, there are some best practices for deployment.

It is important to stop changes from occurring and perform a test deployment to guarantee the success of the production deployment. These steps typically happen during a *maintenance window*. During this time, users should be locked out of the system, so plan this well in advance and during off-peak hours. Deployment is an all-or-nothing event—for example, a single new field on the production organization can make the entire deployment fail if this field does not exist in the deploying organization. Because any changes you make on production during the deployment phase can nullify the final deployment, it is important that no changes occur until deployment finishes.

It is advisable to create a staging environment that allows you to do a test deployment before deploying to production. The staging environment is usually a full-copy sandbox, so that it is as similar to the production organization as possible. For this same reason, you should create or refresh the staging environment during the maintenance window, and not before. Full-copy sandboxes can take some time to create or refresh, so it is important to pad your maintenance window to account for this.

Deployment to the staging environment follows the same procedure as migrating from one development organization to another. This procedure includes manual migration for any component not in the Metadata API, and for any features developed using the Salesforce Web user interface. In addition, it is advisable to manually run all tests in your staging environment to avoid any possible issues before the production deployment. Development environments do not enforce Apex test coverage, but a production organization does.

After you successfully deploy to your staging environment, you need to make some additional changes before deploying to production. If you modified any environmental dependencies in your development environment (for example, to give developers permissions they do not have on production), you need to change those values back to the production values. If you configured service endpoints to test integration with external services, those must also be changed to their production values.

You are now ready to deploy to production. First lock users out of the system, and then perform a Metadata API test deployment on the production organization. This is a validation "check-only" deployment, meaning that the deployment is fully simulated and the success or failure of the deployment is returned, but no components are actually deployed. This step is especially important if there is a time lapse between when the deployment was staged (during normal business hours, for example) and when the actual deployment takes place (when users are locked out over the weekend, for example). If this test deployment is successful, you can deploy changes using the Metadata API or Web interface, depending on the components.

Keeping in mind that all IT organizations are different, the following general procedure recaps the high-level steps you might follow for deploying an enterprise application to a production organization:

1. Announce a maintenance window.
2. Stop all setup changes on production.
3. Create a staging environment.
4. Migrate changes to the staging environment.
5. Change environmental dependencies and services from testing settings to production values.

6. Lock users out of the application.
7. Test deploy using the Metadata API.
8. Deploy to production.
9. Unlock the production organization.

Scheduling the Release

Any time you deploy changes to a production environment your users are directly affected, so it is a good idea to have guidelines for rolling out new functionality. Salesforce.com has over a decade of experience in this area, and you might want to base your rollout procedure on our model:

1. Do not break anything:
   a. Release your production functionality in a test environment first. If you successfully deploy and test in a full-copy sandbox, you can be confident your final deployment to production will succeed.
   b. Back up everything.
   c. Have a fallback plan, just in case.
2. Schedule the release:
   a. Create and announce a maintenance window during which your organization will be unavailable to most users.
   b. Use profiles to control maintenance updates.
3. Inform users of every change:
   a. Create detailed release notes that document new functionality and changes to existing behavior.
   b. Send an email announcing the main features with a link to the release notes.
   c. Create webinars and training sessions to educate users.

Using Profiles to Control Maintenance Updates

During a deployment window, you can use user profiles to limit end user access to the production organization:

1. Create and announce a maintenance window during which your organization will be unavailable to most users:
   - Click **Setup ➤ Manage Users ➤ Mass Email Users** to access an email wizard that allows you to alert all active users about the maintenance window.
2. Use profiles to create and manage the maintenance window:
   - Use the Login Hours related list on user profiles to lock most users out during the maintenance window. Be sure that any system administrator or integration users have access if they need it.
   - Roll out objects, tabs, and apps selectively to different user profiles if you want to allow some users to have access for user acceptance testing.

If your organization includes many profiles, use the following strategy for setting up a maintenance window:

1. Create a new profile named Maintenance with all login hours locked out.
2. Use the Data Loader to extract and save a mapping of users and their user profiles.

3. At the beginning of the maintenance window, use the Data Loader to change all user profiles *except* the administrator's to the Maintenance profile. Note that it is very important to leave login access with the administrator because otherwise all users could remain locked out of the system indefinitely. If any integrations are going to run during the maintenance window, the integration user should also not be locked out.
4. At the end of the maintenance window, use the Data Loader to reload the users with their original profiles.

Platform Glossary

A

Administrator (System Administrator)

One or more individuals in your organization who can configure and customize the application. Users assigned to the System Administrator profile have administrator privileges.

Apex

Force.com Apex code is a strongly-typed, object-oriented programming language that allows developers to execute flow and transaction control statements on the Force.com platform server in conjunction with calls to the Force.com API. Using syntax that looks like Java and acts like database stored procedures, Apex code enables developers to add business logic to most system events, including button clicks, related record updates, and Visualforce pages. Apex scripts can be initiated by Web service requests and from triggers on objects.

App

A collection of components such as tabs, reports, dashboards, and Visualforce pages that address a specific business need. Short for "application." Salesforce provides standard apps such as Sales and Call Center. You can customize the standard apps to match the way you work. In addition, you can package an app and upload it to Force.comAppExchange along with related components such as custom fields, custom tabs, and custom objects. Then, you can make the app available to other Salesforce users from AppExchange.

App Menu

See Force.com App Menu.

AppExchange Directory

A Web directory where hundreds of apps are available to Salesforce customers to review, demo, comment upon, and/or install. Developers can submit their apps for listing on AppExchange if they wish to share them with the community.

AppExchange Upgrades

Upgrading an app is the process of installing a newer version.

Application Programming Interface (API)

The interface that a computer system, library, or application provides in order to allow other computer programs to request services from it and exchange data between them.

B

No Glossary items for this entry.

C

Child Relationship

A relationship that has been defined on an sObject that references another sObject as the "one" side of a one-to-many relationship. For example, contacts, opportunities, and tasks have child relationships with accounts.

See also sObject.

Class, Apex

A template or blueprint from which Apex objects are created. Classes consist of other classes, user-defined methods, variables, exception types, and static initialization code. In most cases, Apex classes are modeled on their counterparts in Java.

Client App

An app that runs outside the Salesforce user interface and uses only the Force.com API—typically running on a desktop or mobile device. These apps treat the platform as a data source, using the development model of whatever tool and platform for which they are designed. See also Composite App and Native App.

Cloud Computing

A virtual development and deployment environment that eliminates the need for computing hardware and software infrastructure and components. Developers and users connect to this environment through a Web browser.

Component, Visualforce

Something that can be added to a Visualforce page with a set of tags, for example, `<apex:detail>`. Visualforce includes a number of standard components, or you can create your own custom components.

Component Reference, Visualforce

A description of the standard and custom Visualforce components that are available in your organization. You can access the component library from the development footer of any Visualforce page or the *Visualforce Developer's Guide*.

Composite App

An app that combines native platform functionality with one or more external Web services, such as Yahoo! Maps. Composite apps allow for more flexibility and integration with other services, but may require running and managing external code. See also Client App and Native App.

Controller, Visualforce

An Apex class that provides a Visualforce page with the data and business logic it needs to run. Visualforce pages can use the standard controllers that come by default with every standard or custom object, or they can use custom controllers.

Custom App

See App.

Custom Field

A field that can be added in addition to the standard fields to customize Salesforce for your organization's needs.

Custom Link

Custom URLs defined by administrators to integrate your Salesforce data with external websites and back-office systems. Formerly known as Web links.

Custom Object

Custom records that allow you to store information unique to your organization.

Custom Report Type

A selection in the Report Wizard that allows you to define the report criteria from which users can run and create custom reports. For example, you may want users to run and customize reports based on accounts with or without opportunities. With custom report types, you can enable users to create reports from predefined standard and custom objects, object relationships, and standard and custom fields that you specify. .

Custom S-Control

Custom Web content for use in custom links. Custom s-controls can contain any type of content that you can display in a browser, for example a Java applet, an Active-X control, an Excel file, or a custom HTML Web form.

Custom View

A display feature that lets you see a specific set of records for a particular object.

D

Dashboard

A visual representation of your custom report data. You can create several dashboard components within a dashboard to give you a real-time snapshot of corporate metrics and key performance indicators.

Database

An organized collection of information. The underlying architecture of the Force.com platform includes a database where your data is stored.

Database Table

A list of information, presented with rows and columns, about the person, thing, or concept you want to track. See also Object.

Data Loader

A Force.com platform tool used to import and export data from your Salesforce organization.

Data Manipulation Language (DML)

An Apex method or operation that inserts, updates, or deletes records from the Force.com platform database.

Dependent Field

Any custom picklist or multi-select picklist field that displays available values based on the value selected in its corresponding controlling field.

Developer Edition

A free, fully-functional Salesforce organization designed for developers to extend, integrate, and develop with the Force.com platform. Developer Edition accounts are available on developer.force.com.

Developer Force

The Developer Force website at developer.force.com provides a full range of resources for platform developers, including sample code, toolkits, an online developer community, and the ability to obtain limited Force.com platform environments.

E

Email Alert

Email alerts are workflow and approval actions that are generated using an email template by a workflow rule or approval process and sent to designated recipients, either Salesforce users or others.

Email Template

A form email that communicates a standard message, such as a welcome letter to new employees or an acknowledgement that a customer service request has been received. Email templates can be personalized with merge fields, and can be written in text, HTML, or custom format.

Enterprise Edition

A Salesforce edition designed for larger, more complex businesses.

Enterprise WSDL

A strongly-typed WSDL for customers who want to build an integration with their Salesforce organization only, or for partners who are using tools like Tibco or webMethods to build integrations that require strong typecasting. The downside of the Enterprise WSDL is that it only works with the schema of a single Salesforce organization because it is bound to all of the unique objects and fields that exist in that organization's data model.

F

Field

A part of an object that holds a specific piece of information, such as a text or currency value.

Field Dependency

A filter that allows you to change the contents of a picklist based on the value of another field.

Field-Level Security

Settings that determine whether fields are hidden, visible, read only, or editable for users based on their profiles. Available in Enterprise, Unlimited, and Developer Editions only.

Folder

A central place to store documents, email templates, and reports. Folders are available for all organizations. Report and document folders are not available in Personal Edition.

Force.com

The salesforce.com platform for building applications in the cloud. Force.com combines a powerful user interface, operating system, and database to allow you to customize and deploy applications in the cloud for your entire enterprise.

Force.com App Menu

A menu that enables users to switch between customizable applications (or "apps") with a single click. The Force.com app menu displays at the top of every page in the user interface.

Force.com IDE

An Eclipse plug-in that allows developers to manage, author, debug and deploy Force.com applications in the Eclipse development environment.

Force.com Migration Tool

A toolkit that allows you to write an Apache Ant build script for migrating Force.com components between a local file system and a Salesforce organization.

Force.com Sites

A feature that allows access to Force.com applications by users outside of a Salesforce organization.

Force.com Web Services API

A Web services-based application programming interface that provides access to your Salesforce organization's information.

Foreign key

A field whose value is the same as the primary key of another table. You can think of a foreign key as a copy of a primary key from another table. A relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

G

Governor limits

Apex execution limits that prevent developers who write inefficient code from monopolizing the resources of other Salesforce users.

Group

A groups is a set of users. Groups can contain individual users, other groups, or the users in a role. Groups can be used to help define sharing access to data or to specify which data to synchronize when using Connect for Outlook or Connect for Lotus Notes.

Users can define their own personal groups. Administrators can create public groups for use by everyone in the organization.

Group Edition

A product designed for small businesses and workgroups with a limited number of users.

H

No Glossary items for this entry.

I

Import Wizard

A tool for importing data into your Salesforce organization, accessible from Setup.

Instance

A server that hosts an organization's data and runs their applications. The Force.com platform runs on multiple instances, but data for any single organization is always consolidated on a single instance.

Integration User

A Salesforce user defined solely for client apps or integrations. Also referred to as the logged-in user in a Force.com Web Services API context.

J

Junction Object

A custom object with two master-detail relationships. Using a custom junction object, you can model a "many-to-many" relationship between two objects. For example, you may have a custom object called "Bug" that relates to the standard case object such that a bug could be related to multiple cases and a case could also be related to multiple bugs.

K

No Glossary items for this entry.

L

Layout

See Page Layout.

Letterhead

Determines the basic attributes of an HTML email template. Users can create a letterhead that includes attributes like background color, logo, font size, and font color.

M

Managed Package

A collection of application components that are posted as a unit on AppExchange, and are associated with a namespace and a License Management Organization. A package must be managed for it to be published publicly on AppExchange, and for it to support upgrades. An organization can create a single managed package that can be downloaded and installed by many different organizations. They differ from unmanaged packages in that some components are locked, allowing the managed package to be upgraded later. Unmanaged packages do not include locked components and cannot be upgraded. In addition, managed packages obfuscate certain components (like Apex) on subscribing organizations, so as to protect the intellectual property of the developer.

Many-to-Many Relationship

A relationship where each side of the relationship can have many children on the other side. Many-to-many relationships are implemented through the use of junction objects.

Master-Detail Relationship

A relationship between two different types of records that associates the records with each other. For example, accounts have a master-detail relationship with opportunities. This type of relationship affects record deletion, security, and makes the lookup relationship field required on the page layout.

Metadata

Information about the structure, appearance, and functionality of an organization and any of its parts. Force.com uses XML to describe metadata.

Metadata-Driven Development

An app development model that allows apps to be defined as declarative "blueprints," with no code required. Apps built on the platform—their data models, objects, forms, workflows, and more—are defined by metadata.

Metadata WSDL

A WSDL for users who want to use the Force.com Metadata API calls.

Multitenancy

An application model where all users and apps share a single, common infrastructure and code base.

MVC (Model-View-Controller)

A design paradigm that deconstructs applications into components that represent data (the model), ways of displaying that data in a user interface (the view), and ways of manipulating that data with business logic (the controller).

N

Native App

An app that is built exclusively with setup (metadata) configuration on Force.com. Native apps do not require any external services or infrastructure.

O

Object

An object allows you to store information in your Salesforce organization. The object is the overall definition of the type of information you are storing. For example, the case object allow you to store information regarding customer inquiries. For each object, your organization will have multiple records that store the information about specific instances of that type of data. For example, you might have a case record to store the information about Joe Smith's training inquiry and another case record to store the information about Mary Johnson's configuration issue.

One-to-Many Relationship

A relationship in which a single object is related to many other objects. For example, an account may have one or more related contacts.

Organization

A deployment of Salesforce with a defined set of licensed users. An organization is the virtual space provided to an individual customer of salesforce.com. Your organization includes all of your data and applications, and is separate from all other organizations.

Organization-Wide Defaults

Settings that allow you to specify the baseline level of data access that a user has in your organization. For example, you can make it so that any user can see any record of a particular object that is enabled in their user profile, but that they need extra permissions to edit one.

Outbound Message

Workflow and approval actions that send the information you specify to an endpoint you designate, such as an external service. An outbound message sends the data in the specified fields in the form of a SOAP message to the endpoint. Outbound messaging is configured in the Salesforce setup menu. Then you must configure the external endpoint. You can create a listener for the messages using the Force.com Web Services API.

Owner

Individual user to which a record (for example, a contact or case) is assigned.

P

Package

A group of Force.com components and applications that are made available to other organizations through the AppExchange. You use packages to bundle an app along with any related components so that you can upload them to Force.com AppExchange together.

Page Layout

The organization of fields, custom links, and related lists on a record detail or edit page. Use page layouts primarily for organizing pages for your users. In Enterprise, Unlimited, and Developer Editions, use field-level security to restrict users' access to specific fields.

Personal Edition

Product designed for individual sales representatives and single users.

Picklist

Selection list of options available for specific fields, for example, the `Industry` field for accounts. Users can choose a single value from a list of options rather than make an entry directly in the field. See also Master Picklist.

Picklist (Multi-Select)

Selection list of options available for specific fields. Multi-select picklists allow users to choose one or more values. Users can choose a value by double clicking on it, or choose additional values from a scrolling list by holding down the Control key while clicking a value and using the arrow icon to move them to the selected box.

Picklist Values

Selections displayed in drop-down lists for particular fields. Some values come predefined, and other values can be changed or defined by an administrator.

Platform Edition

A Salesforce edition based on either Enterprise Edition or Unlimited Edition that does not include any of the standard Salesforce CRM apps, such as Sales or Service & Support.

Primary Key

A relational database concept. Each table in a relational database has a field in which the data value uniquely identifies the record. This field is called the primary key. The relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

Production Organization

A Salesforce organization that has live users accessing data.

Professional Edition

A Salesforce edition designed for businesses who need full-featured CRM functionality.

Profile

Defines a user's permission to perform different functions within Salesforce. For example, the Solution Manager profile gives a user access to create, edit, and delete solutions.

Q

No Glossary items for this entry.

R

Read Only

One of the standard profiles to which a user can be assigned. Read Only users can view and report on information based on their role in the organization. (That is, if the Read Only user is the CEO, they can view all data in the system. If the Read Only user has the role of Western Rep, they can view all data for their role and any role below them in the hierarchy.)

Record

A single instance of a Salesforce object. For example, "John Jones" might be the name of a contact record.

Record Type

A field available for certain records that can include some or all of the standard and custom picklist values for that record. Record types are special fields that you can associate with profiles to make only the included picklist values available to users with that profile.

Record-Level Security

A method of controlling data in which you can allow a particular user to view and edit an object, but then restrict the records that the user is allowed to see.

Relationship

A connection between two objects, used to create related lists in page layouts and detail levels in reports. Matching values in a specified field in both objects are used to link related data; for example, if one object stores data about companies and another object stores data about people, a relationship allows you to find out which people work at the company.

Report

Information you can display or print that provides a summary of your data.

Report Types

Specifies the objects and fields that can be used as the basis of a report. In addition to pre-defined standard report types, you can create custom report types for more advanced reporting requirements.

Running User

The user whose security settings determine what data is displayed in a dashboard. Because only one running user is specified per dashboard, everyone who can access the dashboard sees the same data, regardless of their personal security settings.

S

SaaS

See Software as a Service (SaaS).

S-Control

Custom Web content for use in custom links. Custom s-controls can contain any type of content that you can display in a browser, for example a Java applet, an Active-X control, an Excel file, or a custom HTML Web form.

Attention: S-controls have been superseded by Visualforce pages. After January 2010 salesforce.com will remove the ability to create and distribute new s-controls. Existing s-controls will remain unaffected.

Salesforce IdeaExchange

A forum where salesforce.com customers can suggest new product concepts, promote favorite enhancements, interact with product managers and other customers, and preview what salesforce.com is planning to deliver in future releases. Visit Salesforce IdeaExchange at ideas.salesforce.com.

Salesforce SOA (Service-Oriented Architecture)

A powerful capability of Force.com that allows you to make calls to external Web services from within Apex.

Sandbox Organization

A nearly identical copy of a Salesforce production organization. You can create multiple sandboxes in separate environments for a variety of purposes, such as testing and training, without compromising the data and applications in your production environment.

Session ID

An authentication token that is returned when a user successfully logs in to Salesforce. The Session ID prevents a user from having to log in again every time he or she wants to perform another action in Salesforce. Different from a record ID or Salesforce ID, which are terms for the unique ID of a Salesforce record.

Session Timeout

The amount of time a single session ID remains valid before expiring. While a session is always valid for a user while he or she is working in the Web interface, sessions instantiated via the API expire after the duration of the session timeout, regardless of how many transactions are still taking place.

Setup

An administration area where you can customize and define Force.com applications. Access Setup through the **Setup** link at the top of Salesforce pages.

Sharing

Allowing other users to view or edit information you own. There are different ways to share data:

- Sharing Model—defines the default organization-wide access levels that users have to each other's information and whether to use the hierarchies when determining access to data.
- Role Hierarchy—defines different levels of users such that users at higher levels can view and edit information owned by or shared with users beneath them in the role hierarchy, regardless of the organization-wide sharing model settings.
- Sharing Rules—allow an administrator to specify that all information created by users within a given group or role is automatically shared to the members of another group or role.
- Manual Sharing—allows individual users to share a specific account or opportunity with other users or groups.
- Apex-Managed Sharing—enables developers to programmatically manipulate sharing to support their application's behavior. See Apex-Managed Sharing.

Sharing Model

Behavior defined by your administrator that determines default access by users to different types of records.

Sites

Force.com Sites enables you to create public websites and applications that are directly integrated with your Salesforce organization—without requiring users to log in with a username and password.

SOAP (Simple Object Access Protocol)

A protocol that defines a uniform way of passing XML-encoded data.

Software as a Service (SaaS)

A delivery model where a software application is hosted as a service and provided to customers via the Internet. The SaaS vendor takes responsibility for the daily maintenance, operation, and support of the application and each customer's data. The service alleviates the need for customers to install, configure, and maintain applications with their own hardware, software, and related IT resources. Services can be delivered using the SaaS model to any market segment.

SOQL (Salesforce Object Query Language)

A query language that allows you to construct simple but powerful query strings and to specify the criteria that should be used to select data from the Force.com database.

SOSL (Salesforce Object Search Language)

A query language that allows you to perform text-based searches using the Force.com API.

Standard Object

A built-in object included with the Force.com platform. You can also build custom objects to store information that is unique to your app.

System Log

A separate window console that can be used for debugging code snippets. Enter the code you want to test at the bottom of the window and click Execute. The body of the System Log displays system resource information, such as how long a line took to execute or how many database calls were made. If the code did not run to completion, the console also displays debugging information.

T

Test Method

An Apex class method that verifies whether a particular piece of code is working properly. Test methods take no arguments, commit no data to the database, and can be executed by the `runTests()` system method either through the command line or in an Apex IDE, such as the Force.com IDE.

Translation Workbench

Administration setup area where your users can translate custom field names, picklist values, record types, and page layout sections. The translation workbench also determines which users translate different languages.

Trigger

A piece of Apex that executes before or after records of a particular type are inserted, updated, or deleted from the database. Every trigger runs with a set of context variables that provide access to the records that caused the trigger to fire, and all triggers run in bulk mode—that is, they process several records at once, rather than just one record at a time.

U

Unit Test

A unit is the smallest testable part of an application, usually a method. A unit test operates on that piece of code to make sure it works correctly. See also Test Method.

Unlimited Edition

Unlimited Edition is salesforce.com's flagship solution for maximizing CRM success and extending that success across the entire enterprise through the Force.com platform.

Unmanaged Package

An AppExchange package that cannot be upgraded or controlled by its developer. Unmanaged packages allow you to take any app components and move them "as is" to AppExchange without going through a lengthy publishing process.

URL (Uniform Resource Locator)

The global address of a website, document, or other resource on the Internet. For example, http://www.salesforce.com.

V

Validation Rule

A rule that prevents a record from being saved if it does not meet the standards that are specified.

Visualforce

A simple, tag-based markup language that allows developers to easily define custom pages and components for apps built on the platform. Each tag corresponds to a coarse or fine-grained component, such as a section of a page, a related list, or a field. The components can either be controlled by the same logic that is used in standard Salesforce pages, or developers can associate their own logic with a controller written in Apex.

W

Web Service

A mechanism by which two applications can easily exchange data over the Internet, even if they run on different platforms, are written in different languages, or are geographically remote from each other.

WebService Method

An Apex class method or variable that can be used by external systems, such as an s-control or mash-up with a third-party application. Web service methods must be defined in a global class.

Wizard

A user interface that leads a user through a complex task in multiple steps.

WSDL (Web Services Description Language) File

An XML file that describes the format of messages you send and receive from a Web service. Your development environment's SOAP client uses the Salesforce Enterprise WSDL or Partner WSDL to communicate with Salesforce using the Salesforce Web Services API.

X

XML (Extensible Markup Language)

A markup language that enables the sharing and transportation of structured data. All Force.com components that are retrieved or deployed through the Metadata API are represented by XML definitions.

Y

No Glossary items for this entry.

Z

Zip File

A data compression and archive format.

A collection of files retrieved or deployed by the Metadata API. See also Force.com Project.

Index