1. How is the Force.com Migration tool different than change sets?

The Force.com Migration Tool allows you to migrate metadata between unrelated organizations, so in that sense it's more powerful than change sets. Unlike the Force.com IDE, the Force.com Migration Tool has no graphical user interface. You choose the 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.

1. What is the Production Typical Development Lifecycle?

1). Plan functional requirements.  
2). Develop using Salesforce Web tools, using profiles to hide your changes until they're ready to deploy.  
3). Update profiles to reveal your changes to the appropriate users.  
4). Notify end users of changes.

What is the Sandbox Typical Development 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 the development environment.  
5). Deploy what you've developed to your production organization.

What are some examples of Development Projects?

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

What are the uses for a Sandbox environment?

Use sandboxes for development, testing, training, or other purposes without compromising the data and applications in your Salesforce production org.

What is the easiest way to send configuration changes from one organization to another related organization?

Change Sets

What is the Typical Development Lifecycle when you have an isolated environment for development and one for testing?

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 other environments.  
7). Deploy what you've developed to your production organization.

What is the Typical Development Lifecycle when you have multiple Projects with integration, testing and staging?

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.

What are the 4 Sandbox Types?

--Developer - Developer sandboxes copy customization (metadata), but don't copy production data, into a separate environment for coding and testing.  
--Developer Pro - Developer Pro sandboxes copy customization (metadata), but don't copy production data, into a separate environment for coding and testing. Developer Pro has more storage than a Developer sandbox. It includes a number of Developer sandboxes, depending on the edition of your production organization.  
--Partial Copy - A Partial Copy sandbox is a Developer sandbox plus the data that you define in a sandbox template.  
--Full - Full sandboxes copy your entire production organization and all its data, including standard and custom object records, documents, and attachments. Use the sandbox to code and test changes, and to train your team about the changes. You can refresh a Full sandbox every 29 days.

What is the best type of sandbox to use when testing external integrations?

Full sandbox is best when an external system expects full production data to be present.  
  
\*\*Partial Copy sandboxes may be appropriate in special cases when you want to use sample data or a subset of your actual data. Works well if you're using external IDs.

What is the best type of sandbox to use when performing development?

Developer or Developer Pro sandbox.

What is the best type of sandbox to use when performing User Acceptance Testing (UAT) and Staging?

Full sandbox is best for validation of new applications against production configuration and data.  
  
\*\*Partial Copy sandboxes are appropriate if testing against a subset of production data is acceptable, for example, for regional tests.

What is the best type of sandbox to use when performing production debugging?

Full Sandbox

What is the best type of sandbox to use when performing Unit and Apex tests?

Developer or Developer Pro sandbox

What is the best type of sandbox to use when performing Feature or Regression tests?

Partial Copy sandbox (with a standard data set loaded)

What features are disabled and cannot be enabled in sandboxes?

-Contract expiration warnings  
-Case escalation  
-Contract expiration warnings and case escalation are disabled because they automatically send email to contacts, customers, and production org users.  
-Subscription summary  
-Data exports (by clicking Export Now or Schedule Export on the Weekly Export Service page in Setup)  
-The ability to create Salesforce sandboxes  
-The ability to copy email service addresses that you create in your sandbox to your production org  
-The ability to publish Site.com sites

What are environmental dependencies?

They are settings that are different between a development environment and the production organization.

What are some environmental dependencies?

-Login Privileges  
-Email Addresses  
-Email Recipients  
-External URLs  
-Hard Coded URLs  
-Hard Coded IDs  
-Existing Projects

True/False: Sandboxes has the same number of Licenses as Production.

True

How do you create a User 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.

What should you consider when Activating a Sandbox?

Activating a refreshed sandbox replaces the existing sandbox with the refreshed version, permanently deleting the existing version and all data in it. Your production organization and its data aren't affected.

What is the easiest development tool used by most administrators?

Web Interface (force.com)

What is Force.com IDE?

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

How does Force.com IDE Enable 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 Metadata API.

What are the limits for Force.com IDE?

You can deploy or retrieve up to 10,000 files at once and the maximum size of the deployed or retrieved .zip file is 39 MB. If you need to retrieve or deploy more than either of these limits, you must do so in batches.

What's a good way to structure your project in Forece.com IDE?

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.

How does the Metadata API work?

Metadata API provides two parts that work in conjunction: a rich and powerful metadata model and an application programming interface (API).

What can you do with the MetaData API?

--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 the 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.

What is best tool to use to populate a test environment with large amounts of setup changes?

Force.com Migration Tool

True/False: The Force.com Migration Tool gives you complete control over the retrieve() 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.

True

What is the best tool to use for repetitive deployment using the same parameters—you can retrieve metadata from your organization, make changes, and deploy your changes to an organization?

Force.com Migration Tool

What's a benefit for using the Force.com Migration tool if a highly-technical resource is doing the deployment?

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.

What is the most efficient way to do a multi-stage release process?

Using the Force.com Migration Tool --Scripted retrieval and deployment of components can make this process much more efficient.

What is Data Loader?

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

True/False: You can use excel files in Data Loader.

False

When should use you use Data Loader over other data import tools?

--You need to load 50,000 to 5,000,000 records. ----Data Loader is supported for loads of up to 5 million records. If you need to load more than 5 million records, we recommend you work with a Salesforce partner or visit the App Exchange for a suitable partner product.  
--You need to load into an object that is not yet supported by the import wizards.  
--You want to schedule regular data loads, such as nightly imports.  
--You want to export your data for backup purposes.

What is the most important reason you would need to track Development Changes?

You might have to manually migrate some changes from one organization to another.

How can you ensure that changes can be successfully deployed without overwriting changes on the Production Organization?

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. This may have to be done manually.

What is a Change Process?

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.

What are some best practices for Change Processes?

--Allow no changes in production (most strict)  
--Modify only components in the Metadata API  
--Allow only one administrator to make setup changes (workable if org is small enough for just one admin to handle)  
--Schedule Production Changes

How can you track changes using the Metadata API?

They can be tracked and merged using desktop tools.

How can you track changes using 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

What is a useful tool to track changes manually?

--Tickets  
--AppExchange custom apps such as Change Control  
--SpreadSheets

What are a few data points to record when tracking changes manually?

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

When should you 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.

What is a fundamental difference between traditional Software Development and cloud development?

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.

What is a good way to schedule & categorize a Development Project?

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. A three-tier scheme is a good starting point

What are some categories to use when scheduling a Development Project by judging by where development will take 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.

What are some categories to use when scheduling a Development Project by judging 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.

What is 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.

What is the general process for delivering multiple applications on a release train?

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 ensure that short-term functionality that is delivered to production does not get overwritten when you deploy from your development environment.  
5). Integrate changes and deploy to staging or test environments.  
6). Release to production.

What are a few things that happen during a Salesforce Upgrade?

--New logo—The Salesforce logo is a quick way to verify which version you are using.   
New features—Every release contains new features. --Among those are new components available through the Metadata API.   
--Incremented API version—The API version increments every release, and access to new features requires connecting to the new version of the API.  
--Staggered upgrades—Because your production and sandbox organizations might not be running the same version of the platform during the upgrade window, you might have to wait to deploy components that are new or have additional metadata. If you try to upload a change set that has newer components to an organization that hasn't been upgraded to support them, the deployment will fail.

What does "Migration" mean in the Salesforce context?

Migration is the act of moving configuration changes from one Salesforce organization to another

How can a Migration occur?

Migration can happen in two ways: manually or through the Metadata API.  
--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.  
Metadata migration—Components that are available in the Metadata API can be migrated using desktop tools or change sets.

What are the typical steps to migrate changes?

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 or outbound change set to deploy only a subset of components.  
4). Deploy.

What are some reasons you would NOT was to make changes in Production & then refresh your sandboxes?

--Full Sandboxes can only be refreshed every 29 days  
--any changes on production must be manually migrated in each of your development organizations.  
--component dependencies play a large role in deployment, changes you make on your production organization might prevent you from deploying applications you develop on sandbox.  
--If you have multiple development organizations, you must manually migrate changes from production to sandbox many times.

What is the best way to manage a manual migration?

Establish a change process on your production organization, and to track the changes that require manual migration.

How do you migrate changes from one organization to another using metadata files?

An intermediate tool that interacts with both environments using the Metadata API.

Why do you need a local project to migrate metadata files?

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 don't need to retrieve files unless you want to synchronize with the server.

What are some factors with Deployment Time?

--Number and size of files—The more you 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,changing a field type could require modifying all records that use that field.  
--Test execution—When deploying to a production organization, the number and complexity of Apex tests have a large impact on the deployment time.  
--Network and server availability—These are of minimal concern compared to other factors. However, consider scheduling 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 org during deployment, locking can affect users and the deployment.

What method does the Force.com IDE and the Force.com Migration tool use to deploy changes?

Calling the Metadata API deploy()

True/False: The deploy () method is synchronous.

False

What does the Deployment Status page list?

This page lists all deployments—change sets, Metadata API-based deployments, including deployments started from the Force.com IDE and the Force.com Migration Tool, and package installations.

What does each chart show on the Deployment Status page?

--The first chart shows how many components have already been deployed out of the total and includes the number of components with errors.   
-- A second chart shows how many Apex tests have run out of the total number of tests and the number of errors returned. In addition, the chart shows the name of the currently running test.

What is a deployment validation?

A deployment validation is a deployment that is used only to check the results of deploying components and is rolled back. A validation doesn't save any deployed components or change the Salesforce org in any way. You can determine whether a deployment is a validation only (Validate) or an actual deployment (Deploy) by inspecting the information for pending deployments or the Status column of deployments in the Failed and Succeeded sections.  
  
If a validation completed successfully in the last 10 days, and all tests passed with sufficient code coverage, you can perform a quick deployment by deploying this validation to production without running tests.

How do you Cancel a deployment?

You can cancel a deployment while it's in progress or in the queue by clicking Cancel next to the deployment. The deployment then has the status Cancel Requested until the deployment is completely canceled. A canceled deployment is listed in the Failed section.

How can you have a partially successful deployment?

Only deployments to a non-production org can partially succeed. These are deployments that have the rollbackOnError field set to false in the deployment options and have errors in a subset of components. In a partially succeeded deployment, the failed components aren't committed and the remaining components are committed to the org.

What conditions must be met in order to use the Quick Deployment feature?

--The components have been validated successfully for the target environment within the last 10 days.  
--As part of the validation, Apex tests in the target org have passed.  
--Code coverage requirements are met.  
--If all tests in the org or all local tests are run, overall code coverage is at least 75%, and Apex triggers have some coverage.  
--If specific tests are run with the Run specified tests test level, each class and trigger that was deployed is covered by at least 75% individually.

True/False: You can validate change sets or metadata components through the API or the Force.com Migration Tool.

True

How you can start a quick deployment through Metadata API or the Force.com Migration Tool?

For Metadata API, call deployRecentValidation() and pass it the validation ID. For the Force.com Migration Tool, use the <sf:deployRecentValidation> task.

True/False: If you perform a deployment after a validation, whether through Quick Deploy, a package installation, or a regular deployment, all validations no longer qualify for quick deployment. Re-validate the set of components to quick-deploy.

True

What are some examples of why an Apex test would have slow performance (taking longer than 2 minutes)?

--accessing org data instead of using test data  
--exercising SOQL queries or Apex code with poor performance

What are some best practices for running tests in your deployments to various environments to ensure high-quality deployments in production and shorter deployment times?

--Run local tests in a deployment to a development environment  
--Validate your components before deploying them by performing a deployment validation  
--Use recent validations that were successful in the last 10 days to perform quick deployments  
--Specify the tests to run by using the RunSpecifiedTests test level

What are the different Deployment Dependencies?

--Parent-child—Metadata components may depend on other components.  
--Referenced file—Every file that is referenced by another file must either be included in the deployment plan, or already in the destination organization.  
--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 user interface, the tool enforces the creation of mandatory fields.

Why might you want to divide your deployment into smaller batches?

--The deployment is too large—You can deploy or retrieve up to 10,000 files at once and the maximum size of the deployed or retrieved .zip file is 39 MB.  
--Long deployments—If you experience unusually long deployments, you can divide your deployment into smaller pieces. Smaller deployments 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 by default and those that don't. Testing only those components that require testing speeds deployment and locks fewer components.

What are some ways to determine which components should deploy at the same time if dividing your deployment into smaller batches?

--Deploy components that don't trigger tests—In API version 34.0 and later, the only components that require tests by default are Apex classes and triggers. All other components don't require tests.  
Don't split dependent components—Because file dependencies play such a large role in migrating metadata, it's important not to separate dependent components.  
--Deploy the most numerous components separately  
--Email templates can be deployed separately, but must be deployed before the components that use the templates.  
--Dashboards can be deployed separately, but deploy them before reports in case a custom button links to a report.  
--Reports can contain the largest number of components. They can be deployed after all other components and in multiple batches.

What are the 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.

When is it advisable to edit the package.xml file using the Force.com IDE to deploy?

Only 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.

What are the two deployment targets you can call that will help you determine which components to include in each package.xml when using the Force.com Migration Tool to deploy?

--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.

What are the different processes for renaming a components in different environments?

--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 user interface. This preserves the data in existing records.

How can you delete components using a deployment?

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.

Why would using the AppExchange to Migrate Changes be inefficient?

--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.  
--Managed packages add constraints that make them poorly suited to use as an IT development tool.   
[NOTE: Unmanaged packages can be a useful for distributing initial components to multiple organizations.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.]

What are some best practices for deployments to production?

--It is important to deploy during a period when users will not be making changes to your organization.   
--You should also perform a test deployment to guarantee the success of the production deployment.   
--Remember that Deployment is an all-or-nothing event   
-- Create or refresh the staging environment during the maintenance window that allows you to do a test deployment before deploying to production  
--During the staging test deployment be sure to includes manual migration for any component not in the Metadata API, and for any features developed using the Salesforce user interface.  
--Manually run all tests in your staging environment to avoid any possible issues before the production deployment.

What are the 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.

What is Salesforce's guidelines for rolling out new functionality?

1). Don't 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.

How can you use Profiles to create & manage the maintenance window during a deployment to production?

Edit Login Hours in 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.

What if your organization has many profiles and you would like to lock users out during the 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.

What is the best practice when it comes to bug fixes?

Make all your changes in one place, and then follow a repeatable process for moving the change to production.  
  
If you make the changes in production, there needs to be a process to get those fixes back to the development environments so the bugs are not accidentally re-introduced during the next deployment.

True/False : You can upgrade or downgrade your Salesforce edition.

False

Universal Containers is having problems with developers overwriting code and blaming each other for unit test failures. Because their Salesforce org produces financial reports, their legal counsel has also recommended that they begin auditing changes to the system for compliance reasons.

What recommendation should a Technical Architect make to address both issues?  
  
A). Periodically export Salesforce system audit logs to an offline database  
B). Implement a Source Control system and require developers to commit changes to source control  
C). Require developers to work in separate sandboxes and periodically merge code into a single Software Integration sandbox  
D). Implement a Requirements Traceability Matrix

B). Implement a Source Control system and require developers to commit changes to source control

Universal Containers (UC) has purchased a new application from a partner to enable advanced quoting. UC just hired a developer to help modify the application to meet their needs.

Which two package types should the Architect recommend?  
  
A). Unmanaged package developed in a Partner Developer Edition org  
B). Managed package developed in a Partner Developer Edition org  
C). Unmanaged package developed in a Developer Edition org  
D). Managed package developed in a Developer Edition org

A). Unmanaged package developed in a Partner Developer Edition org  
C). Unmanaged package developed in a Developer Edition org

What consideration should an Architect take into account when asked for a  
recommendation of Agile vs. Waterfall?  
Choose one answer  
  
A). Rapidly changing business requirements.  
B). Project budget and timeframe.  
C). Number of stakeholders involved.  
D). Technical complexity of the project

A). Rapidly changing business requirements.

Universal Containers (UC) has a highly customized org. A Salesforce release is coming up  
soon, and UC has sandboxes that have already been upgraded to the next release.  
What two actions should UC take to ensure that their customizations are not affected by  
the new release?  
Choose two answers  
  
A). Execute all unit tests in the upgraded Sandbox to validate that code has not been  
affected.  
B). Review code in the upgraded Sandbox to ensure that no automatic updates were  
applied to the code.  
C). Testing is not necessary, because Salesforce tests all code and configuration before  
each upgrade.  
D). Review the updated features of the release, and perform functional testing to  
understand the impacts of updated features.

A). Execute all unit tests in the upgraded Sandbox to validate that code has not been  
D). Review the updated features of the release, and perform functional testing to  
understand the impacts of updated features.

A Salesforce release is coming up soon, and Universal Containers (UC) has sandboxes  
that have already been upgraded to the next release. UC has been developing code in an  
upgraded Sandbox, and needs to deploy code to Production which is still on the previous  
release.  
What is required to deploy these changes to Production?  
Choose one answer  
  
A). No adjustments are necessary, because all metadata types will deploy normally to  
Production.  
B). It is not possible to deploy from an upgraded Sandbox to an older Production version.  
C). Versioned metadata needs to be downgraded to a previous API version before  
deployment.  
D). No adjustments are necessary, as long as the deployment is from a Full Sandbox.

C). Versioned metadata needs to be downgraded to a previous API version before  
deployment.