DUMPSBOSS.

Salesforce Certified Development Lifecycle and Deployment Architect

Salesforce Development-Lifecycle-and-Deployment-Architect

Version Demo

Total Demo Questions: 31

Total Premium Questions: 315

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Governance	32
Topic 2, Application Lifecycle Management	58
Topic 3, Testing	57
Topic 4, Change Management	21
Topic 5, Release Management	20
Topic 6, Salesforce Deployment Tools	62
Topic 7, Environments Management	64
Topic 8, Mix Questions	1
Total	315

QUESTION NO: 1

What are three advantages of the package development model?

Choose 3 answers

A. Improving team development and collaboration.

B. Eliminating the need of using change set, which should no longer be used as it can get messy working with package development models.

C. Facilitating automated testing and continuous integration.

D. Significantly reducing the need for manually tracking changes.

E. Providing its own source control, so the source can be deployed In any sandbox orgs.

ANSWER: A C D

Explanation:

The package development model supports modular, source-driven delivery by treating a package as a versioned unit of Salesforce metadata. **Improving team development and collaboration.** is an advantage because teams can organize related functionality into package directories, develop work in parallel, and use a shared version-control workflow to review and integrate changes. **Facilitating automated testing and continuous integration.** is also central to this model: package versions can be created and validated through repeatable command-line or automated build processes, with tests run as part of the pipeline before promotion. Finally, **Significantly reducing the need for manually tracking changes.** is a key benefit because the package version and its declared dependencies provide an explicit, reproducible record of what metadata is delivered. This reduces reliance on manually assembled deployment inventories and makes releases more consistent across environments. Salesforce describes second-generation packaging as a source-driven approach that supports versioning, automation, and modular application development. See the Salesforce second-generation packaging documentation and the Salesforce Packaging Basics Trailhead module.

QUESTION NO: 2

Universal Containers (UC) is considering updating their Salesforce Release Management process. Which three best practices should UC consider for Release Management? Choose 3 answers

A. Design the right sandbox strategy for the release.

B. Release sign-off is only required for Production.

C. Regression testing is mandatory for each release.

D. Maintain a pre/post deployment checklist for each release.

E. Publish a release calendar for each phase of the release.

ANSWER: A D E

Explanation:

Design the right sandbox strategy for the release is a core release-management practice because each sandbox type has different purposes, data capacity, refresh intervals, and deployment uses. Aligning development, integration, testing, and staging activities to suitable environments gives teams controlled promotion paths and reduces the risk of untested work reaching production. Salesforce describes the available sandbox types and their intended use in its Sandbox Overview.

Maintain a pre/post deployment checklist for each release provides a repeatable operational control for release readiness and completion. A checklist can capture required approvals, backup and communication activities, deployment validation, user-impact checks, and post-deployment smoke testing. This ensures that important release steps are performed consistently, even when the release team or scope changes.

Publish a release calendar for each phase of the release is also essential. A visible calendar coordinates development, testing, business validation, deployment windows, change freezes, and stakeholder communications. It lets teams plan dependencies and avoid conflicts with business-critical events. Salesforce recommends planning deployment activities and testing as part of an organized application lifecycle; see Salesforce Application Lifecycle Management.

QUESTION NO: 3

Universal Containers CUC) is looking for advice on how often it should refresh its sandboxes. UC currently uses a development Mfecycle that starts with developer environments and moves to integration testing, QA testing, UAT, and then production. They have many scrum teams working concurrently and the teams do not agree on when refreshes should occur.

What two recommendations should the architect suggest?

Choose 2 answers

A. Sandboxes should be refreshed on the day when the refresh is allowed for that type of sandbox.

B. Production is the only pristine environment.

C. Integration sandboxes should be refreshed rarely because of the burden of maintaining the various API.

D. Development environments should generally be refreshed after each working feature has been successfully migrated.

ANSWER: B C

Explanation:

Production is the only pristine environment. Production is the authoritative source of the organization's current configuration, data, and deployed functionality. A sandbox is a copy made at a specific point in time and will naturally diverge as teams develop, test, and deploy changes. Treating production as the pristine source helps teams establish a consistent baseline for refresh decisions and avoids assuming that any shared nonproduction environment is permanently current or clean.

Integration sandboxes should be refreshed rarely because of the burden of maintaining the various API. An integration environment typically supports coordinated testing among multiple scrum teams and connected systems. Refreshing it can require reestablishing external endpoints, authentication and integration-user access, connected-app settings, test data, and environment-specific configuration. Because that restoration and coordination work can disrupt ongoing integrated testing, refreshes should be planned and infrequent rather than performed merely because the sandbox is eligible. Salesforce documents that sandbox refreshes replace the sandbox contents with a new production copy and that teams must account for post-refresh setup and data considerations. See Salesforce Help: Refresh a Sandbox and Trailhead: Sandbox Strategies.

QUESTION NO: 4

What is the process used to initiate a connection for change sets?

A. Modify the source org to allow an outbound connection to the target org

B. Modify the target org to accept an outbound connection from the source org

C. Modify the target org to accept an inbound connection from the source org

D. Modify the source org to allow an inbound connection to the target org

ANSWER: A C

Explanation:

A change set deployment connection involves the source organization sending an outbound change set and the target organization explicitly authorizing inbound changes from that source. In practice, the source organization is where an

administrator creates and uploads an outbound change set, so it must be enabled and used as the originating organization for the outbound deployment. The target organization establishes the trust boundary by allowing inbound changes from the identified source organization in Deployment Settings. Once that inbound authorization exists, the target becomes available as a destination when the source uploads its outbound change set. This authorization model ensures that metadata cannot be sent to an organization unless that receiving organization has deliberately accepted deployments from the sender. After upload, the change set appears in the target organization's Inbound Change Sets area, where it can be validated and deployed. Salesforce documents that deployment connections for change sets are configured by authorizing inbound changes in the destination organization, while outbound change sets are created and uploaded from the originating organization. See Salesforce Help: Deploy Changes with Change Sets and Trailhead: Create a Change Set and Deploy It.

QUESTION NO: 5

Universal Containers has asked the salesforce architect to establish a governance framework to manage all of those Salesforce initiatives within the company. What is the first step the Architect should take?

A. Implement a comprehensive DevOps framework for all initiatives within Universal Containers

B. Establish a global Center of Excellence to define and manage Salesforce development standards across the organization

C. Identify relevant Stakeholders from within Universal Containers to obtain governance goals and objectives

D. Implement a project management tool to manage all change requests on the project

ANSWER: C

Explanation:

Identify relevant Stakeholders from within Universal Containers to obtain governance goals and objectives is the appropriate first step because governance must be designed around the organization's business strategy, priorities, decision-making needs, risk tolerance, and desired outcomes. The architect should first identify the business and technical groups affected by Salesforce—including executive sponsors, business-unit leaders, product owners, IT, security, compliance, and delivery teams—and use their input to establish a shared understanding of what governance must accomplish. This discovery informs essential framework components such as decision rights, intake and prioritization criteria, funding and ownership models, architecture standards, release policies, and success measures. A governance model imposed before stakeholder goals are understood is unlikely to gain the sponsorship and adoption required to manage initiatives consistently across the enterprise. Salesforce's architect guidance emphasizes engaging stakeholders to understand requirements and establish an effective operating model, while its governance resources describe aligning governance practices with organizational objectives and collaborative decision-making. Once goals, stakeholders, and accountabilities are clear, Universal Containers can select supporting structures and processes, such as a Center of Excellence, DevOps practices, and demand-management tooling. See Salesforce's governance guidance and Salesforce Center of Excellence basics.

QUESTION NO: 6

Universal Containers is in the final stages of building a new application to track custom containers. During a review of the application, a business subject Matter Expert mentioned that it would be nice to be able to track additional container types beyond what was originally scoped during the plan and design phase. Which two actions should be performed to mitigate the risk? Choose 2 answers.

A. Escalate and communicate to stakeholders the risk and mitigate it by allocating additional resources to support the new requirement based on stakeholdersâ€™ input.

B. Have a discussion with the business subject Matter Expert and communicate that the Salesforce has limitations in supporting such a feature to mitigate the risk.

C. Escalate and communicate to stakeholders the risk and mitigate it by extending the timeline of the project to support the new requirement based on stakeholdersâ€™ input.

D. Have a discussion with the business subject Matter Expert and communicate that anew developer environment will be needed to mitigate the risk.

ANSWER: A

Explanation:

Escalate and communicate to stakeholders the risk and mitigate it by allocating additional resources to support the new requirement based on stakeholdersâ€™ input. This is the appropriate risk-management action because the request introduces scope after the planned design has been substantially completed. The delivery team should document the potential impact, raise it through the established project governance process, and obtain stakeholder direction on whether the added container types are sufficiently valuable to warrant a change. If stakeholders approve the scope change, allocating suitably skilled resources can reduce delivery risk by enabling the team to complete the additional analysis, configuration, development, testing, and deployment work without displacing essential work already committed to the release. This approach maintains transparency about the trade-offs involved and ensures that the business—not an individual contributor—owns the decision to accept the cost and impact of the new requirement. Salesforce's application lifecycle guidance emphasizes defining release processes and coordinating changes across teams, while its governance guidance stresses shared decision-making, prioritization, and alignment with business outcomes. See Salesforce Release Management and Salesforce IT Governance.

QUESTION NO: 7

What two things are needed to delete metadata with a deploy() call?= Choose 2 answers

A. Package.XML file.

B. The CURRENT API version must be used.

C. DestructiveChanges.xml file.

D. PurgeOnDelete option must be set to TRUE.

ANSWER: A C

Explanation:

To delete metadata through a Metadata API `deploy()` operation, the deployment ZIP must include both `package.xml` and `destructiveChanges.xml`. The `package.xml` manifest establishes the deployment package and specifies the API version used for the request. The separate destructive-change manifest identifies the metadata components to remove, organized by metadata type and member name. For example, it can list obsolete custom objects, fields, Apex classes, or other supported components that are intended for deletion. Salesforce processes the deployment as a server-side metadata operation, so the deletion request must be packaged and submitted in this manifest-based form rather than issued as an ad hoc deletion command. The destructive changes manifest is used alongside the package manifest in the deployment archive, allowing the deployment to both describe the package and explicitly identify the metadata targeted for removal. Salesforce's Metadata API documentation describes deployment as submitting a ZIP file containing manifest information, while the Metadata API Developer Guide documents destructive changes as part of the deployment package process. See Deploy Metadata and Delete Components from an Organization.

QUESTION NO: 8

Universal Containers (UC) had added a Service team to the Salesforce Platform. The

Service team would like to have a few dozen of the service centers entered into the system as

technical reference data. The service centers are made searchable in many different web forms

and rather independent from all other business entities. In the past, they had to manually add any

new service centers in each sandbox in the code migration path, they would like to eliminate the

manual work if it is possible.

What is an optimal way to accomplish this requirement?

A. Add the service centers to a hierarchical custom settings.

B. Add the service centers to a list custom settings.

C. Define a brand-new custom object with a picklist field to host all of the service centers.

D. Add all of the service centers to a custom metadata type.

ANSWER: D

Explanation:

Add all of the service centers to a custom metadata type. Custom metadata types are designed for application configuration and technical reference data that must be consistently available across development, test, and production environments. Each service center can be represented as a custom metadata record, with fields for the values needed by the web forms, such as a display name, identifier, geographic information, or an active indicator. The records are metadata rather than ordinary business data, so they can be included in source control and deployed through the same change-management process as the associated object definitions, flows, Apex, and user-interface components. This eliminates the need to manually recreate the reference entries in every sandbox along the promotion path. Custom metadata records can also be queried from Apex and accessed by declarative tools, allowing the forms to search and present a consistent set of service centers without requiring relationships to other business records. Salesforce specifically identifies custom metadata types as suitable for configuration data that is packageable and deployable, including both the type definition and its records. See Salesforce Developer Guide: Custom Metadata Types and Salesforce Help: Custom Metadata Types Overview.

QUESTION NO: 9

Universal Containers has a long-running Salesforce project that will be released in six months. A separate support team must deploy low-risk production fixes every week. Both teams use source control. What branching strategy should an Architect recommend?

A. Maintain a production branch for weekly fixes and merge approved fixes into the long-running project branch.

B. Commit all weekly fixes only to the long-running project branch.

C. Make weekly fixes directly in production and synchronize source control after the project release.

D. Freeze all production fixes until the long-running project is complete.

ANSWER: A

Explanation:

Maintain a production branch for weekly fixes and merge approved fixes into the long-running project branch. This approach isolates the support team's production-bound changes from incomplete project functionality while ensuring that every production correction is incorporated into the future release. The production branch represents the current live baseline, and the project branch can continue to evolve independently.

Each weekly fix should be developed, reviewed, tested, and deployed from a short-lived fix branch created from the production baseline. After deployment, the approved change must be merged forward into the long-running project branch to avoid regression when the project is eventually released. This practice supports controlled parallel development and consistent source history. See Salesforce guidance on Application Lifecycle Management.

QUESTION NO: 10

Universal Containers (UC) has two subsidiaries which operate independently. UC has made the decision to operate two of separate Salesforce orgs, one for each subsidiary. However, certain functions and processes between the two orgs must be standardized. Which two approaches should UC take to develop customizations once, and make them available in both orgs? Choose 2 answers

A. Develop the functionality in a sandbox and deploy it to both production orgs

B. Set up Salesforce-to-Salesforce to deploy the functionality from one org to the other

C. Create a managed package in a sandbox and deploy it to both production orgs

D. Create a package in a Developer Edition org and deploy it to both production orgs

ANSWER: D

Explanation:

Create a package in a Developer Edition org and deploy it to both production orgs provides a reusable distribution mechanism for shared customizations. A dedicated Developer Edition org can act as the common development and packaging org, where UC builds the metadata that must remain standardized, such as custom objects, fields, Lightning components, Apex, flows, permissions, and other supported package components. UC can then distribute the same package to each subsidiary's production org, ensuring that each receives an identical, versioned set of shared functionality.

This approach also establishes a clear ownership boundary for common capabilities: enhancements are developed and tested centrally, packaged as a release artifact, and installed in each target org. Packages are intended to group and distribute related metadata, making them appropriate when the same customization needs to be delivered to multiple Salesforce orgs. For an unmanaged package, recipients can modify installed components after installation; a managed package is generally used when the publisher needs stronger versioning and upgrade control. Salesforce's packaging overview is available in the Salesforce Packaging Developer Guide, and the package creation process is covered in Salesforce Trailhead: Create an Unmanaged Package.

QUESTION NO: 11

Universal containers is looking to install a new application to enable advanced quoting in its current Professional Edition org. The org is near capacity with object and tab limits. Which two solutions should the Architect recommend? Choose 2 answers

A. Install an Aloha certified App

B. Upgrade to an Enterprise Edition org

C. Create and install an unmanaged package

D. Buy more user licenses to increase org limits

ANSWER: A B

Explanation:

Installing an Aloha certified App is appropriate because Aloha-designated AppExchange solutions are designed for Professional Edition organizations and their package components do not consume the customer organization's custom object and custom tab limits. This allows Universal Containers to add the required advanced-quoting capability while avoiding the immediate capacity constraint that would otherwise prevent installation of an application with additional objects and tabs. The AppExchange provides package and edition compatibility information that should be validated as part of selection and procurement: Salesforce AppExchange.

Upgrading to an Enterprise Edition org is also appropriate because Enterprise Edition provides materially greater customization and configuration capacity than Professional Edition, including higher applicable limits and broader platform capabilities. This gives the organization sufficient headroom not only for an advanced quoting solution, but also for its related data model, user interface components, automation, security configuration, and future growth. An architect should assess edition capabilities and limits before committing to the upgrade, since edition choice affects both available features and organizational scalability. Salesforce publishes edition-level feature and limit information in its product documentation: Salesforce Editions and Features.

QUESTION NO: 12

Universal Containers is having trouble deploying metadata from SIT to UAT. UAT is

complaining that it does not recognize some new Salesforce metadata types to be deployed. The deployment from Dev to SIT worked perfectly

What could be the problem?

A. There is no problem, this is expected behavior.

B. UAT is on a preview release and SIT is not.

C. SIT is on a preview release and UAT is not.

D. Use the DX command line instead.

ANSWER: C

Explanation:

SIT is on a preview release and UAT is not. This release-version mismatch explains why metadata can deploy successfully from Dev to SIT but fail when promoted to UAT. Salesforce preview sandboxes are upgraded to the upcoming seasonal release before production and non-preview sandboxes. Therefore, SIT can recognize metadata types, metadata attributes, or platform capabilities introduced in that upcoming release. If UAT remains on the current release, its Metadata API does not yet know those types and cannot process a deployment containing them.

For a release-aligned deployment path, environments that exchange metadata should support the same Salesforce release and feature set. During the preview window, teams should either ensure that downstream validation environments are also preview instances when appropriate, defer deployment of metadata dependent on the new release, or wait until the non-preview environment is upgraded. Salesforce publishes the preview schedule and identifies which sandbox instances are upgraded early in its Release Calendar. The platform's deployment process relies on the target organization being able to recognize and validate the metadata included in the deployment, as described in the Metadata API deployment documentation.

QUESTION NO: 13

5. Universal Containers (UC) is planning to move to Salesforce Sales Cloud and retire its homegrown on-premise system. As part of the project, UC will need to migrate 5 million Accounts, 10 million Contacts, and 5 million Leads to Salesforce.

Which three areas should be tested as part of data migration?

Choose 3 answers

A. Lead assignment

B. Data transformation against source system

C. Contact association with correct Account

D. Account and Lead ownership

E. Page layout assignments

ANSWER: B C D

Explanation:

Data migration testing must validate that the migrated dataset is both accurate and usable in the target Salesforce org. **Data transformation against source system** is essential because source values commonly require cleansing, normalization, mapping, conversion, deduplication, or defaulting before they comply with Salesforce field definitions and business requirements. Testing compares transformed target data to the source and approved transformation rules, including record counts and field-level results.

Contact association with correct Account must be tested because Contacts depend on correct parent Account relationships. A successful load count is insufficient if Contacts are connected to the wrong Accounts or left without the

intended relationship. Validation should confirm relationship mapping through external IDs or other migration keys and reconcile expected relationship totals.

Account and Lead ownership is also a core migration validation area. Ownership determines record visibility, sharing behavior, responsibility, and downstream reporting. The migration team should verify that each Account and Lead is assigned to the intended active Salesforce user or queue, and that any source-to-target owner mapping and exception handling produce the expected results. Salesforce's data migration guidance emphasizes planning data mapping, cleansing, loading, and validation, while relationship and ownership values are standard record data that must be correctly established during loading. See Salesforce Data Migration and Salesforce Data Loader documentation.

QUESTION NO: 14

Which two options should be considered when making production changes in a highly regulated and audited environment?

Choose 2 answers

A. All changes including hotfixes should be reviewed against security principles.

B. Any production change should have explicit stakeholder approval.

C. No manual steps should be carried out.

D. After deployment, the development team should test and verify functionality in production.

ANSWER: A B

Explanation:

All changes including hotfixes should be reviewed against security principles because urgent remediation does not remove the organization's obligation to protect data, preserve least-privilege access, prevent unintended exposure, and assess the security impact of configuration or code changes. A consistent review process also creates evidence that security controls were considered for every production release, including expedited work. Salesforce's security guidance emphasizes incorporating security into architecture, development, and operational practices rather than treating it as a separate, final activity. See the Salesforce Well-Architected security guidance.

Any production change should have explicit stakeholder approval because regulated release processes require clear accountability, authorization, and an auditable decision record before a change reaches production. The appropriate business, technical, security, and compliance stakeholders can confirm that the change has acceptable risk, meets business requirements, and follows the organization's documented change-control process. Salesforce Setup Audit Trail supports governance by recording many configuration changes and helping organizations investigate administrative activity. Organizations can use this operational evidence alongside their approval and deployment records to demonstrate controlled production change management. Refer to Salesforce Setup Audit Trail documentation.

QUESTION NO: 15

Universal Containers CUC) Customer Community is scheduled to go live in the Europe, Middle East, and Africa (EMEA) region in 3 months. UC follows a typical centralized governance model. Two weeks ago, the project stakeholders informed the project team about the recent changes in mandatory compliance requirements needed to go live. The project team analyzed the requirements and have estimated additional budget needs of 30^0 of the project cost for incorporating the compliance requirements.

Which management team is empowered to approve this additional budget requirements?

A. Security Review Committee

B. Project Management Committee

C. Executive Steering Committee

D. Change Control Board

ANSWER: C

Explanation:

Executive Steering Committee is the appropriate authority to approve this additional budget because the compliance change has a material financial and delivery impact: it requires funding equal to 30% of the original project cost and affects a scheduled regional go-live. In a centralized governance model, the executive steering body provides strategic oversight, owns major investment decisions, resolves escalated risks and cross-functional conflicts, and authorizes changes that exceed the project team's delegated authority. Mandatory compliance requirements also require an executive-level decision because the organization must balance regulatory obligations, customer-community launch commitments, available funding, and the potential need to adjust scope, schedule, or resources. The committee can formally approve the incremental investment and ensure the project remains aligned with business priorities and enterprise risk tolerance. Salesforce's governance guidance emphasizes clear decision-making structures and executive sponsorship for initiatives that affect organizational priorities and outcomes. See Salesforce's Well-Architected guidance and its change-management resources for principles supporting executive governance, sponsorship, and organizational alignment.

QUESTION NO: 16

All AppExchange products are subject to Salesforce security reviews.

What is the most common reason that the prospect AppExchange products fail the security review?

A. Cross-site scripting

B. CRUD/FLS (field level security)

C. Session hacking

D. SOQL injection

ANSWER: B

Explanation:

CRUD/FLS (field level security) is the most common AppExchange security-review issue because an application must enforce each user's object- and field-level access when it reads, creates, updates, or deletes Salesforce data. Apex commonly runs in system mode, which can bypass the permissions that the current user would normally have in the user interface. Consequently, managed-package code must deliberately apply the appropriate access controls rather than assuming that a query or DML operation is safe merely because it succeeds in Apex.

Salesforce recommends using user-mode database operations where appropriate, or enforcing object and field permissions through mechanisms such as `WITH USER_MODE`, `Security.stripInaccessible()`, and describe-based checks. These controls help ensure that package functionality does not expose restricted fields or allow a user to modify records and objects beyond their assigned permissions. Proper CRUD and FLS enforcement is therefore a foundational requirement for securely packaging an AppExchange product and for passing the Salesforce security review. See Salesforce's Apex data security guidance and the AppExchange security review documentation.

QUESTION NO: 17

Which are two key benefits of fully integrating an agile issue tracker with software testing and continuous integration tools? Choose 2 answers?

A. Developers can see automated test statuses post code commit on a specific user story.

B. Developers can collaborate and communicate effectively on specific user stories.

C. Developers can observe their team velocity on the burn chart report in the agile tool.

D. Developers can use the committed code's build status directly on the user story record.

ANSWER: A D

Explanation:

“Developers can see automated test statuses post code commit on a specific user story.” is a key benefit because an integrated delivery toolchain connects the work item to the validation performed for the change. Developers, testers, and release stakeholders can quickly confirm whether the automated tests associated with a committed change passed, failed, or require investigation, while retaining traceability to the requirement that prompted the work. This shortens feedback cycles and supports informed release decisions.

“Developers can use the committed code's build status directly on the user story record.” is also a key benefit. Continuous integration automatically builds and validates committed code; exposing that build outcome in the issue tracker makes the delivery state visible where planning and development work is managed. A team can therefore track a story from implementation through automated build and test validation without manually copying status information between systems. This integration supports the Salesforce DX emphasis on automated testing and continuous integration, as described in Salesforce DX App Development and Salesforce’s guidance on continuous integration and continuous delivery.

QUESTION NO: 18

Universal Containers CUC) is embarked on a large Salesforce transformation journey, UC's DevOps team raised a question about tracking Salesforce metadata throughout the development lifecycle across sandboxes all the way to production.

As the deployment architect of the project, what should be the recommendation to track which version of each feature in different environments?

A. Use an Excel sheet to track deployment steps and document the SFDX commands.

B. Use an AppExchange or third-party tool that is specialized in Salesforce deployment.

C. Use ChangeSet to track deployed customizations.

D. Use Salesforce SFDX commands to deploy to different sandboxes.

E. Use a version control system, such as Git, as the source of truth for Salesforce metadata and feature versions across environments.

ANSWER: E

Explanation:

Use a version control system, such as Git, to store Salesforce metadata in source format and establish a definitive history for every feature. Each feature or work item can be developed in an isolated branch, reviewed through pull requests, and merged into controlled release branches. Commits and tags provide an auditable record of precisely which metadata version is associated with each release, while branch and release-management practices identify what has been promoted through integration, test, staging, and production environments. This approach supports repeatable deployments because the metadata deployed to an environment is retrieved from a known repository revision rather than being inferred from changes made directly in an org. Salesforce DX source tracking complements this model during development by identifying local and org changes, but the repository remains the durable system of record for the lifecycle state of a feature. A Git-based workflow also enables automated validation and deployment pipelines, code review, rollback planning, and reliable collaboration among multiple developers. Salesforce recommends organizing metadata as source and using version control as part of a modern development model; see the Salesforce development models guidance and the Salesforce Trailhead Git fundamentals.

QUESTION NO: 19

Universal Containers has a complex deployment coming up. The deployment will include several Apex classes which depend on custom settings that hold important configuration. How should an Architect manage this deployment?

A. Script the deployment of all functionality via the Force.com Migration Tool

B. Manually deploy and populate custom settings in production using a change set

C. Create a custom metadata type and include this in your deployment to production

D. Manually deploy and populate the custom settings in production prior to the Apex Class deployment

ANSWER: C

Explanation:

Create a custom metadata type and include this in your deployment to production is the appropriate approach because custom metadata is designed for application configuration that must move predictably through the development lifecycle. Custom metadata type definitions and their records are metadata, so they can be version controlled, included in source-driven deployment workflows, and promoted between sandbox and production alongside the Apex code that uses them. This provides a repeatable, auditable deployment process and ensures that the configuration required by the classes is present when the release is deployed.

For a complex release, the architect should model deployable configuration as custom metadata records and have Apex query that metadata as needed. Salesforce supports deploying custom metadata records through its metadata deployment mechanisms, including change sets and the Metadata API. This avoids treating environment configuration as a separate, manual post-deployment activity and makes release contents more complete and consistent across environments. Where appropriate, protected custom metadata can also help package developers control which configuration values subscribers can modify. See Salesforce's Custom Metadata Types overview and Metadata API documentation for custom metadata.

QUESTION NO: 20

Which two decisions should be made by an Architecture Review Board (ARB)? Choose 2 answers

A. Whether to create a new Salesforce object or override an existing object using a new Record Type

B. Whether to utilize the Waterfall or Agile methodology on the project

C. What testing tools should be used to track integration testing requirements

D. Whether to implement Single Sign -On with SAML or delegated authentication

ANSWER: A D

Explanation:

An Architecture Review Board should make decisions that establish durable, enterprise-wide technical patterns and guardrails. Whether to create a new Salesforce object or use an existing object with a new Record Type is a core data-model architecture decision. It affects the organization's canonical data structure, relationships, security model, reporting approach, integrations, and long-term maintainability. Record Types allow different business processes, picklist values, and page layouts for records of the same object, so selecting that approach rather than introducing a separate object requires architectural oversight. Salesforce describes Record Types and their role in presenting different business processes and values in its RecordType reference.

Whether to implement Single Sign-On with SAML or delegated authentication is also an enterprise architecture and security decision. It determines how Salesforce authenticates users, how identity is integrated with the organization's identity provider, and the operational and security responsibilities shared between systems. SAML SSO uses assertions from an identity provider to establish authentication in Salesforce, while delegated authentication uses an external service to validate credentials. These choices need consistent standards across the Salesforce environment and therefore belong under ARB governance. Salesforce documents the SAML SSO model and configuration considerations in its Single Sign-On with SAML documentation.

QUESTION NO: 21

Universal Containers CUC) is embarked on a large Salesforce transformation journey, UC ' s DevOps team raised a question about tracking Salesforce metadata throughout the development lifecycle across sandboxes all the way to production.

As the deployment architect of the project, what should be the recommendation to track which version of each feature in different environments?

A. Use an Excel sheet to track deployment steps and document the SFDX commands.

B. Use an AppExchange or third-party tool that is specialized in Salesforce deployment.

C. Use Change Set to track deployed customizations.

D. Use Salesforce SFDX commands to deploy to different sandboxes.

E. Use a source-control repository, such as Git, to version Salesforce metadata and track feature promotion across environments.

ANSWER: E

Explanation:

Use a source-control repository, such as Git, as the authoritative record of Salesforce metadata and feature versions. Each feature should be represented by metadata in source format and committed to version control through a branch and pull-request strategy. Commits, tags, and release branches provide an auditable history of exactly what changed, when it changed, and which approved version was promoted to each environment. The delivery pipeline can then deploy the same immutable commit or release tag through integration, test, staging, and production environments, while recording deployment results and environment status. This approach supports collaborative development, code review, automated validation, traceability, rollback to a known prior version, and reliable release reproducibility. Salesforce DX is designed around this source-driven development model: retrieve and convert org metadata into source format, keep it under version control, and use automated deployment processes to promote it. A deployment-management product may integrate with this repository and enhance orchestration or reporting, but it does not replace source control as the system of record for feature versions. Salesforce recommends using version control as part of its development lifecycle and CI practices. See Salesforce source control guidance and Salesforce continuous integration guidance.

QUESTION NO: 22

The opportunity Service and opportunity Service Test classes are in package A but are used only in package B. Both second-generation packages have the same namespace. Therefore, they should be moved to package B for better organization and control.

What should the architect recommend for this process?

A. Set the classes as deprecated in package A and recreate them in package B.

B. Move the classes of package A to package B and change the code for package B that called this class from package A.

C. Move the classes of package A to package B and create new package versions.

D. Set the classes as deprecated in package A and recreate them in package B with new names.

ANSWER: B

Explanation:

Move the classes of package A to package B and change the code for package B that called this class from package A. This approach properly realigns component ownership with the package that consumes the functionality. Because both second-generation packages use the same namespace, the classes can be transferred as part of the source and package-version management process rather than duplicated under a new namespace. The team should remove the classes from package A's package source, add them to package B's source, and create package versions that reflect the revised ownership. Any package dependencies and references affected by the change must be updated so that package B compiles and installs against the intended package composition. The deployment sequence should also be planned carefully: package B must include the moved classes before a package A version removes them, preventing dependency or upgrade issues for subscribers. Salesforce DX supports organizing a project into multiple package directories and creating second-generation package versions from the source assigned to each package. See Salesforce's second-generation packaging

documentation and its guidance on package dependencies for the required package-version and dependency-management considerations.

QUESTION NO: 23

What are three advantages of using a Source Control system alongside a multi -sandbox development strategy? Choose 3 answers

A. Perform code reviews before promoting to a pre -production sandbox

B. Automatically deploy changes from sandbox to production

C. Keep a history of changes made by each developer

D. Create a branching strategy that tracks each feature or change separately

E. Act as a backup in case of catastrophic data loss

ANSWER: A C D

Explanation:

Using source control as the shared system of record for a multi-sandbox Salesforce development model enables teams to collaborate safely and maintain a dependable audit trail. “Perform code reviews before promoting to a pre -production sandbox” is an important advantage because developers can submit changes through pull requests, allowing peers to review the source, discuss implementation details, and approve it before the changes are merged into the branch used for integration or preproduction deployment. “Keep a history of changes made by each developer” is also fundamental: commits preserve who made a change, when it was made, and the associated file-level differences, making troubleshooting, rollback, and auditing much more manageable. Finally, “Create a branching strategy that tracks each feature or change separately” supports parallel work across multiple sandboxes. Feature branches isolate in-progress work, reduce collisions between developers, and provide a controlled path for merging completed work into shared branches. Salesforce recommends integrating source control into the development lifecycle and using version-control practices to manage work and collaboration across environments. See the Salesforce DX source control guidance and Salesforce’s DevOps architecture guidance.

QUESTION NO: 24

Universal Containers (UC) has two major releases every year and the team always run into longer deployment times. In which 2 ways can UC reduce deployment time? Choose 2 answers?

A. Use recent deployment validations and the quick deploy feature.

B. Deploy components in groups to reduce deployment time.

C. Specify the test to run by using RunSpecifiedTests test level.

D. Validate the deployment before migrating components to production.

ANSWER: A C

Explanation:

Using recent deployment validations and the quick deploy feature reduces production deployment duration because Salesforce lets a successful validation be reused for a Quick Deploy within the permitted time window. The validation has already performed the required checks and tests, so the subsequent production deployment can avoid repeating that work. This is especially valuable for planned releases: the team can validate the exact deployment package in advance, resolve any issues, and then use Quick Deploy during the release window. Salesforce documents the eligibility requirements and timing for this approach in its deployment guidance.

Specifying the tests to run with the RunSpecifiedTests test level can also reduce deployment time when the deployment uses the Metadata API or Salesforce CLI and the team explicitly supplies the relevant test classes. Rather than running the

full suite of local tests, the deployment executes only the named tests while still requiring those tests to pass and meeting the production code-coverage requirement. This supports a focused, well-maintained regression-test strategy for release artifacts. Salesforce describes the available test levels, including RunSpecifiedTests, in the Metadata API deploy() documentation.

QUESTION NO: 25

As a part of technical debt cleanup project, a large list of metadata components has been identified by the business analysts at Universal Containers for removal from the Salesforce org. How should an Architect manage these deletions across sandbox environments and production with minimal impact on other work streams?

A. Generate a destructivechanges.xml file and deploy the package via the Force.com Migration Tool

B. Perform deletes manually in a sandbox and then deploy a Change Set to production

C. Assign business analysts to perform the deletes and split up the work between them

D. Delete the components in production and then refresh all sandboxes to receive the changes

ANSWER: A

Explanation:

Generate a destructivechanges.xml file and deploy the package via the Force.com Migration Tool is correct because metadata deletions should be managed as a repeatable, source-controlled deployment activity rather than as individual manual changes. A destructive changes manifest explicitly identifies the metadata types and component names to remove. The same deployment artifact can be validated and executed in each sandbox and then promoted to production through the established release process. This creates an auditable, consistent deletion path and avoids disrupting unrelated work that may be occurring in parallel environments.

Using the Metadata API deployment mechanism also allows the team to test the deletion package in lower environments, confirm dependencies and regression-test affected functionality before production, and coordinate the cleanup with normal release governance. Salesforce supports destructive changes in Metadata API deployments; the deployment package includes a manifest describing the components to delete, along with the appropriate package manifest for the deployment. This approach is particularly suitable for a large inventory of obsolete metadata because it is automatable and can be versioned alongside the organization's configuration source.

See Salesforce's Metadata API guide for deleting components and the Metadata API deployment documentation.

QUESTION NO: 26

Universal Containers CUC) has decided to improve the quality of work by the development teams. As part of the effort, UC has acquired some code review software licenses to help the developers with code quality.

Which are two recommended practices to follow when conducting secure code reviews? Choose 2 answers

A. Generate a code review checklist to ensure consistency between reviews and different reviewers.

B. Focus on the aggregated reviews to save time and effort, to remove the need to continuously monitor each meaningful change.

C. Conduct a review that combines human efforts and automatic checks by the tool to detect all flaws.

D. Use the code review software as the tool to flag which developer has committed the errors, so the developer can improve.

ANSWER: A C

Explanation:

Generate a code review checklist to ensure consistency between reviews and different reviewers. is a recommended secure-review practice because a defined checklist makes security expectations repeatable and helps reviewers consistently

assess common risk areas. For Salesforce development, this can include checks for CRUD and field-level security enforcement, secure sharing behavior, appropriate use of dynamic SOQL, avoidance of exposed secrets, and correct input validation. A checklist supports a documented, scalable review process even when different developers or reviewers participate.

Conduct a review that combines human efforts and automatic checks by the tool to detect all flaws. reflects the recommended defense-in-depth approach of pairing automated static-analysis and quality tooling with human review. Automated checks efficiently identify known patterns, insecure constructs, and violations of defined rules at scale. Human reviewers contribute application context, assess authorization and business-logic implications, and determine whether the implementation safely meets its intended behavior. Used together, these practices help teams find a broader range of defects and security issues before deployment. Salesforce's secure coding guidance emphasizes reviewing code for security vulnerabilities and applying platform security controls consistently. See Salesforce Code Review guidance and Salesforce Code Analyzer documentation.

QUESTION NO: 27

Universal Containers (UC) has many different business units, all requesting new projects to be built into a single Salesforce Org. UC management is concerned with a lack of appropriate project properties and roadmap for the Salesforce ecosystem. What should an Architect recommend?

A. Use design Standards for Governance.

B. Create a Center of Excellence with a charter document.

C. Create a Release Management Process.

D. Create project charters for each project.

ANSWER: B

Explanation:

Create a Center of Excellence with a charter document. A Salesforce Center of Excellence provides the operating model needed when multiple business units share one org and compete for delivery capacity. It establishes a cross-functional governance body that represents business and technology stakeholders, evaluates incoming demand against agreed strategic objectives, and prioritizes work across the enterprise rather than allowing projects to proceed independently. The charter formally defines the group's scope, decision rights, membership, accountability, intake and prioritization approach, success measures, and escalation paths. These elements give management a consistent way to determine which initiatives are appropriate investments and to maintain an integrated, transparent roadmap for the Salesforce ecosystem.

The Center of Excellence also creates a sustainable structure for balancing innovation, technical health, platform standards, security, and ongoing operational needs. By making ownership and portfolio decisions explicit, it helps ensure that the shared org evolves according to enterprise priorities and a coherent long-term architecture. Salesforce describes the role and value of this model in its Center of Excellence Basics Trailhead module and its Center of Excellence decision guide.

QUESTION NO: 28

Universal Containers has multiple project learns building into single org. The project teams are concerned with design conflicts and ensuring a common design process What should an Architect recommend to prevent this conflict?

A. Create a Center of Excellence Charter document.

B. Create Design Standard for Governance.

C. Create a backup system using GIT Repositories.

D. Create a Release Management process.

ANSWER: B

Explanation:

Create Design Standard for Governance is correct because a shared set of design standards gives every project team an agreed framework for making architecture and implementation decisions in the same Salesforce org. The standards can define approved patterns for data modeling, integration, security, automation, naming, metadata ownership, API use, and exception handling. They should also establish a design-review process, clear decision rights, required documentation, and an escalation path for deviations. This enables teams to identify cross-project dependencies early and ensures that solutions are evaluated against common architectural principles rather than being designed independently in ways that conflict later. Governance is particularly important in a single-org model because teams share metadata, platform limits, data, integrations, and release dependencies. A practical standards process is maintained by appropriate architecture and governance stakeholders, communicated to delivery teams, and applied throughout design and implementation—not merely at deployment time. Salesforce's architecture guidance emphasizes governance as a means to align decision-making, manage technical consistency, and support scalable change across the organization. See the Salesforce Well-Architected Governance guidance and the Salesforce Architect Journey Trailhead module for related architecture-governance principles.

QUESTION NO: 29

Universal Containers uses multiple Salesforce orgs for its different lines of business (LOBs). In a recent analysis, the architect found that UC could have a more complete view of its customers by gathering customer data from different orgs.

What two options can an architect recommend to accomplish the customer 360-degree view?

Choose 2 answers

A. Implement a Complete Graph multi-org strategy by allowing each org to connect directly to every other, reading and writing customer data
from the orgs where it has been originally created.

B. Migrate from multi-org to single-org strategy, consolidating customer data in the process.

C. Implement a Single Package multi-org strategy by developing and deploying to all orgs a managed package which reads and consolidates
customer 360-degree view from the different orgs.

D. Implement a Hub-and-Spoke multi-org strategy by consolidating customer data In a single org, which will be the master of customer data,
and using integration strategies to let the LOBs orgs read and write from it.

ANSWER: B D

Explanation:

Migrating from a multi-org to a single-org strategy can provide a customer 360-degree view because customer records, related activity, and supporting business data are consolidated into one Salesforce data model. A unified org can establish common identity, duplicate-management, security, sharing, reporting, and data-governance practices, making a complete customer profile available without requiring users or applications to assemble it from separate orgs. This approach is appropriate when the lines of business can operate under shared processes and governance.

Implementing a Hub-and-Spoke multi-org strategy also supports a customer 360-degree view when separate LOB orgs must remain in place. The hub acts as the authoritative master for customer data, while integrations synchronize or expose the governed customer information to the spoke orgs. Defining the hub as the system of record allows UC to apply consistent matching, survivorship, ownership, and data-quality rules while giving LOBs controlled read and write access through integration. This pattern preserves organizational separation where needed while creating a centrally governed customer profile. Salesforce architecture guidance discusses choosing between single- and multi-org models in its multi-org architecture guidance and integration considerations in the Salesforce Integration Patterns overview.

QUESTION NO: 30

Universal Containers are using Salesforce for Order Management and has integrated with an in-house ERP system for order fulfillment. There is an urgent requirement to include a new order status value from the ERP system into the Order Status pick list in Salesforce. Which are two considerations when addressing the above requirement? Choose 2 answers

A. Existing Apex test classes may start falling in Production.

B. Implement the change in the sandbox, validate, and release to Production.

C. The change can be performed in Production, as it is a configuration change.

D. Integration with the ERP system may not function as expected.

ANSWER: B D

Explanation:

“Implement the change in the sandbox, validate, and release to Production.” is appropriate because a picklist-value addition is still a metadata/configuration change that should follow the organization’s release process. A sandbox provides an environment to validate the new Order Status value, confirm its availability for the appropriate record types, run regression testing, and package or deploy the tested metadata through the approved path. This is especially important for an urgent request, because urgency does not remove the need for controlled validation and traceability.

“Integration with the ERP system may not function as expected.” is also a key consideration. The ERP integration may use an explicit transformation or mapping between ERP status codes and Salesforce Order Status values. The new ERP value must be added to that mapping and validated end to end, including inbound payload processing, API user permissions, error handling, and downstream automation that responds to an order-status update. Testing the complete integration flow before release ensures that Salesforce accepts the new value and that fulfillment updates continue to be processed reliably. Salesforce recommends using sandboxes and deployment tools to test and promote changes between environments. See Salesforce Change Sets and Salesforce REST API Developer Guide.

QUESTION NO: 31

Universal Containers has multiple project teams integrating Salesforce to various systems Integration Architects are complaining about the various integration patterns used by the teams and lack a common understanding of the integration landscape. What should architect recommended to address the challenges?

A. Implement a data governance policy and publish the documentation to all teams.

B. Recommend an outbound message design pattern to be used for all teams.

C. Recommend a fire-and-forget design pattern to be used for all teams.

D. Create design standards focused on integration and provide training to all teams

ANSWER: D

Explanation:

Create design standards focused on integration and provide training to all teams is correct because the stated challenge is organizational consistency: teams are selecting patterns independently and do not share a clear view of the enterprise integration landscape. Integration design standards establish a common decision framework for choosing appropriate interaction patterns, such as request-response, event-driven, batch, or process integration, based on requirements including latency, reliability, volume, ownership, security, and error handling. Standards can also define shared conventions for interface contracts, authentication, observability, versioning, retry behavior, and documentation. Training makes those standards actionable by ensuring architects and delivery teams understand how to apply them consistently during solution design and implementation. This approach creates durable governance without assuming that one technical pattern is appropriate for every integration use case. Salesforce’s architecture guidance emphasizes selecting integration patterns based on business and technical requirements and using an enterprise integration strategy to promote consistency. See Salesforce Integration Patterns and Salesforce Integration Architecture.