Common Customer Requirement Calibration Guide for SDP and Competency applications

Table of Contents

Introduction

This technical calibration guide is developed for AWS partners who have applied, or are interested in, the Amazon Web Services (AWS) Competency or Service Delivery program (SDP). This guide covers the “Common Customer Example Requirements” in all the designation checklists. Unique requirements for specific designations will be addressed in separate documents.

The calibration guide format is the FAQs for each control. It provides clarity on the expected level of details of the requested evidence and improve the application quality. It helps partners reduce application cycle time and achieve Competency or SDP designation faster. Additionally, partners can adopt the best practices in this technical guide to improve their AWS service offerings.

Each control has the following FAQs:

Why is this important?
This section explains why a particular control is essential from a well-architected perspective.

What are the criteria for passing this control?
This section discusses what level of information is needed to pass a particular control. It clarifies the requirement for partner to self-assess when collecting artifacts.

Technical Enablement Resources
This section discusses how to implement the specific control using AWS services. It provides step-by-step guidance and different options for partner to adopt AWS recommended best practices.

What are good example responses?
This section provides good response examples that meet the control and displays the level of depth and expertise required in the assessment.

What are unacceptable responses?
This section is composed of response examples not meeting the requirement of the control.

DOC-001

Requirement

Provide Architecture diagram designed with scalability and high availability:

AWS Partner must submit architecture diagrams depicting the overall design and deployment of its AWS Partner solution on AWS as well as any other relevant details of the solution for the specific customer in question.

The submitted diagrams are intended to provide context to the AWS Solutions Architect conducting the Technical Validation. It is critical to provide clear diagrams with an appropriate level of detail that enable the AWS Solutions Architect to validate the other requirements listed below.

Each architecture diagram must show:

- All of the AWS services used.
- How the AWS services are deployed, including virtual private clouds (VPCs), availability zones, subnets, and connections to systems outside of AWS.
- Elements deployed outside of AWS, e.g. on-premises components, or hardware devices.
- How design scales automatically - Solution adapts to changes in demand. The architecture uses services that automatically scale such as Amazon S3, Amazon CloudFront, AWS Auto Scaling, and AWS Lambda.
- How design has high availability with multi-AZ or multi-region deployment. When intentional tradeoffs have been made (e.g. to optimize cost in favor of high availability), please explain the customer's requirements.

Criteria for Passing

- An architecture diagram depicting the overall design and deployment of your solution on AWS that includes ALL architecture diagram requirements listed above.
- Explanation of how the major solutions elements will keep running in case of failure/outage.
- Description of how the major solutions elements scale up automatically.

Why is this important?

Having a well-defined architecture helps you describe the high-level scope of the engagement to customer. High availability and scalability helps customer reduce business risks.

Technical Enablement Resources

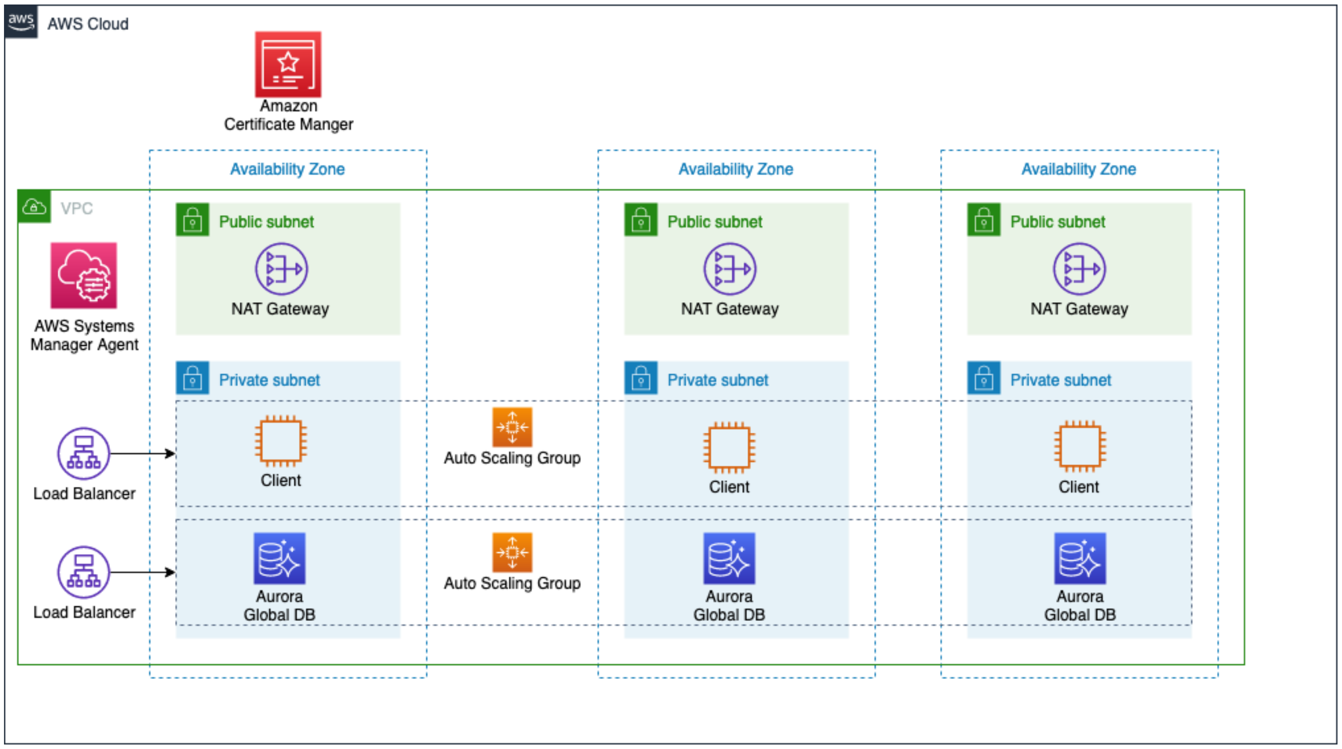
How can you implement this?

- If you do not have a Architecture Diagram, use a tool such as draw.io to create a diagram. You may also use other tools of your choice and import our set of architectural images from our toolkits. It's important that the diagrams indicate primary AWS native services, demonstrate major networking components (such as private and public subnets, regions, and availability zones), and how components communicate with users over the internet.

Resources

- Architectural ToolKit
- AWS Architecture Center

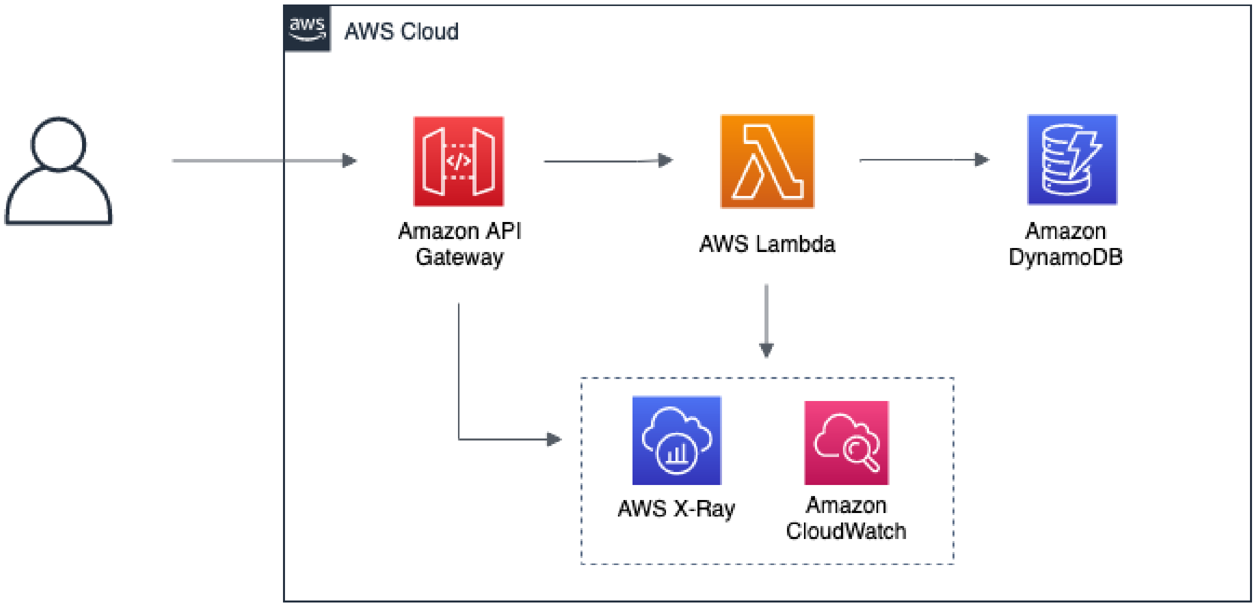
Good Example Response #1: Classic 3-tier Web App



Here, we show a highly available architecture that spans three Availability Zones, which contain Auto Scaling Groups of EC2 instances to easily scale up as well as to protect the application in case of failure/outage.

- ***Note:*** All of the services are in the appropriate place in terms of networking (i.e. NAT gateway in public subnets, ACM located outside the VPC, AWS Systems Manager Agent inside VPC, etc).

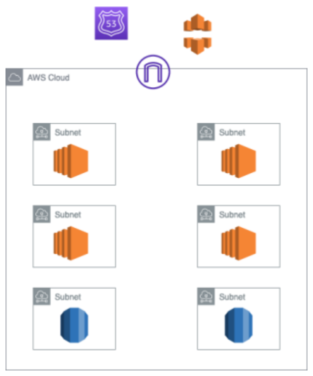
Good Example Response #2: Serverless Application



Here, we have a fully serverless architecture, so the traditional region/AZ/subnet networking is not required. This demonstrates a client accessing the application via API requests through Amazon API Gateway, where the request moves to a number of Lambda functions that handles the request (POST, PUT, UPDATE, DELETE), and completes the action in the Amazon Dynamo DB. Amazon CloudWatch along with AWS X-Ray log and trace the web traffic through the application for observability purposes. Due to the serverless nature of the application, all services scale automatically and prevent failure/outage from impacting the application health. We review the service quotas associated with AWS services with customers to ensure architecture can scale appropriately to their business needs.

Unacceptable/Incomplete Response

- Missing key AWS native services or misplaced services within VPC/regions, no clarity on how the services combine to form the application, or how traffic travels through the application. Also, a lack of description surrounding major solution elements running in case of failure/outage and lack of addressing how application scales up automatically.



ACCT-001

Requirement

Define Secure AWS Account Governance Best Practice:

AWS expects all Services Partners to be prepared to create AWS accounts and implement basic security best practices. Even if most of your customer engagements do not require this, you should be prepared in the event you work with a customer who needs you to create new accounts for them.

Establish internal processes regarding how to create AWS accounts on behalf of customers when needed, including:

- When to use root account for workload activities.
- Enable MFA on root.
- Set the contact information to corporate email address or phone number.
- Enable CloudTrail logs in all region and protect CloudTrail logs from accidental deletion with a dedicated S3 bucket.

Criteria for Passing

- Documents describing Security engagement SOPs which met all the 4 criteria defined above (Acceptable evidence types are security training documents, internal wikis, or standard operating procedures documents).
- Description of how Secure AWS Account Governance is implemented in one (1) of the submitted customer examples.

Why is this important?

Not every one of your customer engagements will require creating AWS accounts, but we need our Partners to be equipped with this knowledge and have a consistent way to onboard both consultants and customers. Consequences of not doing this right can be very detrimental. We have had Partners create accounts for customers using their personal email account and when they left the company, customers were stuck not being able to perform any root user activities, and also were exposed to a huge security risk.

Technical Enablement Resources

How can you implement this?

- From the AWS Prescriptive Guidance page, follow the specific topic below to Secure Your Account
    - ACCT.01 – Set account-level contacts to valid email distribution lists
    - ACCT.02 – Restrict use of the root user (also addresses setting up MFA for root user)
    - ACCT.03 – Configure console access for each user
    - ACCT.07 – Deliver CloudTrail logs to a protected S3 bucket

Additional Resources

- Setting S3 Bucket Policy for Multiple accounts
- Turn on CloudTrail in Additional Accounts
- CloudTrail Log Files from Multiple Accounts

Additional Recommendations

- The following examples are acceptable uses of the root user (but not all encompassing):
    - Restore IAM user permissions, activate IAM access to the billing console.
    - Close your AWS account, change your AWS Support plan, changing your account settings.
    - Configure an Amazon S3 bucket to enable MFA.

Good Customer Example Response

- Root user has MFA enabled, and is only used for tasks that require it (example - recently we used root to upgrade our AWS Support plan). All of our contact information is set to company owned phone numbers, and the contact email addresses are set to an internal distribution list instead of a single email address. We have CloudTrail enabled in all AWS regions (not only the ones we actively use), and the logs are sent to a dedicated S3 bucket that has MFA delete set up to protect against accidental deletion - the bucket is only accessible by 3 individuals within the company (CISO, Lead Architect, and the Dev Manager) and is restricted by IAM policies. We have attached to the application a PDF of our Security Engagement SOP that discusses these standard best practices.

Unacceptable/Incomplete Response

Multiple examples of unacceptable responses:

- Stating information contrary to the best practices (ex. - contact information isn't set to a group internally, CloudTrail only enabled in specific regions, CloudTrail logs not being sent to dedicated bucket with limited access, etc).
- Only submitting a simple self-assessment description - the SOP is required (external links must be immediately accessible to PSA conducting the review).
- Partner responding "N/A" or an answer similar to any of the following: "The whole operational activity was provided by the customer team.", "We did not perform this as part of this engagement, this was the customer's responsibility", etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

ACCT-002

Requirement

Define identity security best practice on how to access customer environment by leveraging IAM:

- Define standard approach to access customer-owned AWS accounts, including:
    - Both AWS Management Console access and programmatic access using the AWS Command Line Interface or other custom tools.
    - When and how to use temporary credentials such as IAM roles.
    - Leverage customer's existing enterprise user identities and their credentials to access AWS services through Identity Federation or migrating to AWS Managed Active Directory.
- Establish best practices around AWS Identity and Access Management (IAM) and other identity and access management systems, including:
    - IAM principals are only granted the minimum privileges necessary. Wildcards in Action and Resource elements should be avoided as much as possible.
    - Every AWS Partner individual who accesses an AWS account must do so using dedicated credentials.

Criteria for Passing

- Security engagement SOPs which met the 2 criteria defined above (Standard approach to access customer owned accounts and establishing best practices around IAM). Acceptable evidence types are: security training documents, internal wikis, standard operating procedures documents. A written description in the self-assessment excel is not acceptable.
- We recognize every customer engagement is different and you may have scenarios where customers only give out IAM users. **Only one example** is required to demonstrate how IAM best practices are implemented.

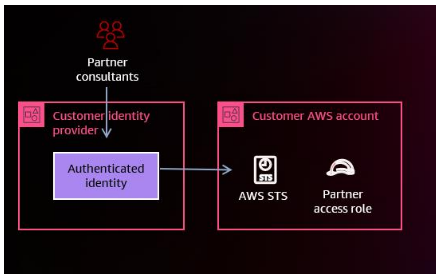
Why is this important?

Having a default, standard recommendation for accessing customer accounts is critical to earn customer trust. As customers' trusted advisor, you should educate and help customers implement AWS IAM security best practices. Following the principle of least privilege for AWS principals limits the risk of data breaches, unauthorized use of services, and minimizes attack surfaces and risk. With these standard IAM best practices in place, you can quickly onboard new customers and consultants.

Technical Enablement Resources

How can you implement this?

- The number one most important principle is to avoid using static credentials such as long-term access keys in IAM. Instead, you can work with your customers to create IAM roles and generate temporary security credentials to access their AWS resources. For machine identities, such as EC2 instances or Lambda functions, require the use of IAM roles instead of IAM users with long term access keys. For human/workforce identities, use AWS SSO or federation with IAM, to access AWS accounts. Specifically, we recommend Partners use federation with an identity provider for a human user to access customer AWS account using temporary credentials. Below is an example diagram.



- When setting permissions with IAM policies, you should grant only the permissions required to perform a task. You do this by defining *least-privilege permissions*. You can leverage IAM Access Analyzer to generate least-privilege policies based on access activity.

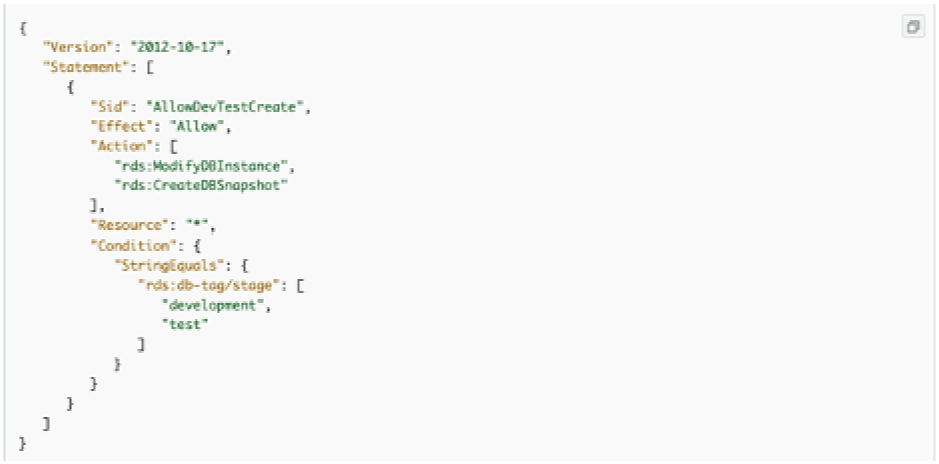
Additional Resources

- Require identities to dynamically acquire temporary credentials.
- Regularly review IAM permissions.
- See IAM Access Analyzer policy validation.

Good Example Response

- PDF, accessible link, or file is attached on the application that clearly documents the Security engagement SOP.
- The customer example should include information similar to the following:
    - Our solution had 3 main components - S3 buckets, EC2 WebServers and RDS databases. Our solution had the following roles created:
      — WebServerEC2S3Role: role created to grant WebServers full read and write access to S3 buckets.
      — EC2RDSRole: role created to grant WebServers full read and write access to RDS instances.
      — AdminRole: had access to create, delete, modify, read and write to all resources and accounts in our organization.
      — DeveloperRole: has access to create, delete select services in their account, but were not given permissions to do so in the production account.
      — SupportRole: Had the ability to read most resources across the organization, except resources which security and compliance information.

    - Attaching a policy that demonstrates principle of least privilege:



```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowDevTestCreate",
         "Effect": "Allow",
         "Action": [
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:db-tag/stage": [
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

    - Every developer user signs-in using their own Active Directory credential, which is integrated with AWS using SAML to assume a role defined by the client's security team.

Unacceptable/Incomplete Response

- A written description in the self-assessment checklist in lieu of an attached SOP.
- An attached policy that includes wildcards in "Action" or very broad permissions.
- Anything making a reference to use of long-term credential usage or shared credentials.
- Partner responding "N/A" or an answer similar to any of the following: "The whole operational activity was provided by the customer team.", "We did not perform this as part of this engagement, this was the customer's responsibility", etc. **While we understand not every engagement may require this, partners must have these**

policies in place. These policies must be followed for at least one (1) of the case studies submitted.

OPE-001

Requirement

Define, monitor and analyze customer workload health KPIs:

AWS Partner has defined metrics for determining the health of each component of the workload and provided the customer with guidance on how to detect operational events based on these metrics.

Establish the capability to run, monitor and improve operational procedure by:

- Defining, collecting and analyzing workload health metrics w/ AWS services or 3rd Party tool.
- Exporting standard application logs that capture errors and aid in troubleshooting and response to operational events.
- Defining threshold of operational metrics to generate alert for any issues.

Criteria for Passing

- Standardized documents or guidance on how to develop customer workload health KPIs with the three components above.
- Description of how workload health KPIs are implemented in one (1) of the submitted customer examples.

Why is this important?

By understanding the health of your workload and operations, you can help customers learn how to address risk and incident response. They can determine when the appropriate responses are needed. Without these capabilities, debugging an error will be difficult and could cost the customer significant time and effort to address performance issues.

Technical Enablement Resources

How can you implement this?

- Designing and Implementing Logging and Monitoring with Amazon CloudWatch
- Work with your customer to establish business goals for the workload, and ensure the customer is educated on the KPIs tracking and alerts.
- On AWS, you can use distributed tracing services, such as AWS X-Ray, to collect and record traces as transactions travel through your workload, generate maps to see how

transactions flow across your workload and services, gain insight to the relationships between components, and identify and analyze issues in real time.
- Establish patterns of workload activity to identify anomalous behavior so that you can respond appropriately if necessary. You can use CloudWatch Anomaly Detection feature, which applies statistical and machine learning algorithms to metrics of systems and applications. It helps to determine normal baselines and surface anomalies with minimal user intervention.

Good Example Response

- PDF, accessible link, screen shot of dashboard, or file is attached on the application that clearly documents how to develop customer workload health KPIs.
- Our application runs primarily with Lambda functions which upload files into S3 buckets. For these functions we monitor duration, billed duration, memory size, max memory used and errors, dead-letter errors, invocations and iterator age, concurrent executions, unreserved concurrent executions and throttles. These logs and events are aggregated in CloudWatch, where we have several dashboards to visualize this (attached dashboard snapshot). To understand overall flow of transactions we have traceId's using Python's AWS X-Ray SDK for each of our incoming requests.

Unacceptable/Incomplete Response

- No standardized documentation or guidance provided.
- Generalized answer such as: "We use CloudWatch to monitor the application and set alarms for various metrics" without providing relevant details for the targeted designation of the service offering (i.e.- RDS, Redshift, DevOps Competency, etc.).
- Partner responding "N/A" or an answer similar to any of the following: "The whole operational activity was provided by the customer team.", "We did not perform this as part of this engagement, this was the customer's responsibility", etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

OPE-002

Requirement

Define a customer runbook/playbook to guide operational tasks:

- Create a runbook to document routine activities and guide issue resolution process with a list of operational tasks and troubleshooting scenarios covered that specifically addresses the KPI metrics defined in OPE-001.

Criteria for Passing

- Standardized documents or runbook met the criteria defined above.

Why is this important?

It’s important to guide your customers for normal operational activity, and enable your customers on the desired state of workloads - including risks, clear steps on how to remediate common issues, and how to manage the flow of change in your environment. Without this information conveyed successfully to your customers, there may be a degradation in service quality, and an accumulation of mis-configurations, which may pose serious security risk to their system. Ensure a successful delivery outcome by providing them with all of this information and train them on the same.

Technical Enablement Resources

How can you implement this?

- Work with your customers to define operational scenarios & tasks that need to be covered in a runbook to ensure customers are receiving documentation and training that will enable their responses to events in the future.
- Ensure there is enough trained personnel on the customer side to support troubleshooting, and train or recommend adjustments of personnel as necessary to maintain effective support.
- Create the runbook that addresses incident response and root cause analysis to specific, well-understood events. When possible and appropriate, implement runbooks as code to ensure consistent, speedy responses, as well as a reduction of errors that may be caused from manual processes.

Additional Resources

- Runbooks - Well-Architected Framework
- Creating runbook automation using AWS Systems Manager

Good Example Response

- A PDF, accessible link, or file is attached on the application to the runbook or standardized documents.

Unacceptable/Incomplete Response

- Link that is unable to be accessed by the PSA conducting the evaluation

- A short description written in the self-assessment checklist w/o demo of typical scenarios
- Partner responding "N/A" or an answer similar to any of the following: "The whole operational activity was provided by the customer team.", "We did not perform this as part of this engagement, this was the customer's responsibility", etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

OPE-003

Requirement

Use consistent processes (e.g. checklist) to assess deployment readiness:

Deployments are tested or otherwise validated before being applied to the production environment. For example, DevOps pipelines used for the project for provisioning resources or releasing software and applications.

Use a consistent approach to deploy to customers including:

- A well-defined testing process before launching in production environment.
- Automated testing components.

Criteria for Passing

- A deployment checklist example or written description met all the criteria defined above.

Why is this important?

Test and validate changes to help limit and detect errors. Automate testing to reduce errors caused by manual processes, and reduce the level of effort to test.

Technical Enablement Resources

How can you implement this?

- Ask yourself a few of the questions listed in this reference here (Operational Readiness Review Questions) to help get started building your own Operational Readiness Reviews (ORR) program.
- Incorporate critical components for deployment readiness check:
    - Use version control to manage code and deployment assets.

    - Develop a standard procedures for testing and validating changes in non-production environments before deploying to production.
    - Establish a system for managing approvals of changes before being deployed to production.

Good Example Response

- PDF, screen shot, or word doc attached to the initial application that contains the deployment checklist
- Customer example should include a thorough explanation of the testing procedures, for example:
    - This solution has a robust set of environments with development, multiple QA environments, and then AWS staging and production environments.
    - To make changes to a specific environment (let's call it EnvA), we alter the CloudFormation templates and then apply them in the lowest environment (dev) first, before deploying the application via Jenkins CI to EnvA where we run a set of automated functional tests written in Ruby with Selenium Webdriver and the Cucumber Watir test framework (which is pre-configured with Capybara) to ensure that infrastructure changes do not break the applications. Then we apply the CloudFormation templates to staging, and retest there.
    - We run load tests in the staging environment with JMeter, to ensure that infrastructure changes do not reduce the performance of the application, with the staging and production environments configured as similarly as possible to allow this. If infrastructure changes work in staging, we re-deploy the templates to production.
    - Application changes are initially tested in the dev environment, then deployed to QA via a Jenkins CI process, and once a release is ready, it is first deployed to staging where load testing happens, and only after it passes a load test does the team initiate a production deploy. This combination of functional and non-functional testing helps us validate that infrastructure changes are valid.

Unacceptable/Incomplete Response

- A lack of PDF, screenshot, etc attached that contains the deployment checklist
- An extremely brief description of the testing process, which only addresses a small number of steps, such as "We always approach our solutions with AWS Well-Architected framework in mind for deployment, architecture, or delivery. We have a standard test and deployment process".
- Partner responding "N/A" or an answer similar to any of the following: "The whole operational activity was provided by the customer team.", "We did not perform this as part of this engagement, this was the customer's responsibility", etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

NETSEC-001

Requirement

Define security best practices for Virtual Private Cloud (Amazon VPC) and other network security considerations:

Establish internal processes regarding how to secure traffic within VPC, including:

- Security Groups to restrict traffic between Internet and Amazon VPC
- Security Groups to restrict traffic within the Amazon VPC
- Network ACL to restrict inbound and outbound traffic

Criteria for Passing

- Written description/documents on network security best practices met the criteria defined above.
- Description of how network security is implementation in one (1) of the submitted customer examples.

Why is this important?

Security groups control the inbound and outbound traffic of the instance(s) in your or your customers' AWS accounts. Network Access Control List (NACL) allows or denies specific inbound or outbound traffic at the subnet level. If these are not configured properly, your AWS resources could be exposed to undesired traffic. These two services are a great method for tightening the overall security of your VPC.

Technical Enablement Resources

How can you implement this?

- Trace the data flow within your AWS workload. For each set of resources, ensure that you only permit the minimal required protocols to communicate with the expected source resources. You can add Network ACLs as an additional layer of defense. Use recommended network security best practices to configure security groups and NACL rules for common VPC scenarios.
- You can use Firewall Manager to centrally manage security groups including:
    1. Configure common baseline security groups across your organization,
    2. Audit existing security groups in your organization, and
    3. Get reports on non-compliant resources and remediate them.

Additional Resources

- Control traffic to resources using security groups
- Control traffic to subnets using Network ACLs
- Best practices to control traffic at all layers

Good Example Response

- Attached documents or description on network security best practices that include the details of security group and Network ACLs. For example, all clients do not have open ports for anonymous sources and are only allowed to use security group references or specific IP ranges.
- The one (1) customer example should include the following information:
    - Description of all security groups and connections within the architecture, including if it connects the VPC to the internet, connects items within the VPC.
    - A description of all NACL's within the architecture to aid in restricting inbound or outbound traffic, and the purpose for each.
    - A screenshot of an example security group for a major component of the architecture.
- A thorough explanation of any data store that is not located within a private subnet.

Unacceptable/Incomplete Response

- Not addressing data stores that are visibly unsecured via subnets in the architecture diagram.
- Inbound traffic allowed from 0.0.0.0/0 without restriction to any portion of the architecture aside from the front end (or specific pieces that are intended to be fully public facing/accessible).
- Partner responding "N/A" or an answer similar to any of the following: "The whole operational activity was provided by the customer team.", "We did not perform this as part of this engagement, this was the customer's responsibility", etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

NETSEC-002

Requirement

Define data encryption policy for data at rest and in transit:

Establish internal processes regarding a data encryption policy used across all customer projects:

- Summary of any endpoints exposed to the Internet and how traffic is encrypted.
- Summary of processes that make requests to external endpoints over the Internet and how traffic is encrypted.
- Enforcing encryption at rest. By default, you should enable the native encryption features in an AWS service that stores data unless there is a reason not to.
- All cryptographic keys are stored and managed using a dedicated key management solution.

Criteria for Passing

- Provide the data encryption and key management policy that meets the criteria defined above.
- Description of how data encryption is implementation in one (1) of the submitted customer examples.

Why is this important?

Un-encrypted data in transit runs the risk of being intercepted and viewed by unauthorized entities. Defining a data encryption approach that includes the storage, rotation, and access control of keys, you can protect customer content against unauthorized users and unnecessary exposure to authorized users.

Technical Enablement Resources

How can you implement this?

- Identify encryption requirements and adopt encryption best practices to help customers meet organizational, legal, and compliance requirements. You can follow specific recommendations below:
    - **Enforce encryption in transit:** You can use AWS services HTTPS endpoints when communicating via APIs. HTTP requests can also be automatically redirected to HTTPS in Amazon CloudFront or on an Application Load Balancer.
    - **Implement secure key management:** Use AWS Key Management Service (KMS) to manage your encryption keys that are used to protect customer data. Additionally, you can use AWS CloudHSM to meet compliance requirements programs such as FIPS 140-2 Level 3,
    - **Implement certificate management:** Use AWS Certificate Manager (ACM) to manage certificates securely. You can offload the termination of Secure Sockets Layer (SSL)/Transport Layer Security (TLS) connections to managed services such as Elastic Load Balancing or Amazon CloudFront and utilize ACM to store and distribute private keys and certificates.

Additional Resources

- Protecting Data in Transit
- Protecting Data at Rest
- Issuing and Managing Certificates

Good Example Response

- Attach a policy (PDF, etc) that addresses all of the requirements listed above.
- Customer example must include details similar to the following:
    - None of our lambda microservices have exposed endpoints, all microservices are fronted by API Gateway with mutual SSL/TLS authentication and a CloudFront Distribution with TLS 1.2 enabled HTTPS traffic. The certificates are managed through AWS Certificate Manager. (add snapshot of each service example)
    - We use KMS with all services (list specific ones in the architecture diagram) that support it. SSL certificates were issued by AWS ACM and attached to ALB SSL listeners. Load balancers use the certificate to decrypt the traffic flowing into the servers where the application is deployed. RDS uses an AWS KMS to encrypt these resources. S3 buckets are encrypted via Amazon S3-Managed Keys (SSE-S3) (add snapshot of each service example)

Unacceptable/Incomplete Response

- Generic statement without connecting to the specific resources/workload components:
    - Stating “We use AWS KMS for managing the sensitive keys, and ALL EBS volumes are encrypted using KMS”.
    - Not addressing all components: “We use SSL for the workspace endpoint URL”.
- Partner responding “N/A” or an answer similar to any of the following: “The whole operational activity was provided by the customer team.”, “We did not perform this as part of this engagement, this was the customer’s responsibility”, etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

REL-001

Requirement

Automate Development and leverage infrastructure-as-code tools:

Changes to infrastructure are automated for customer implementation

- Tools like AWS CloudFormation, the AWS CLI, or other scripting tools were used for automation.
- Changes to the production environment were not done using the AWS Management Console.

Criteria for Passing

- Written description of deployment automation **and** an example template (e.g., CloudFormation templates, architecture diagram for CI/CD pipeline) meet the criteria defined above.

Why is this important?

Automating infrastructure limits the misconfiguration risk and improves productivity. It enables consistent deployment to customers which also allows partners to scale the solution.

Technical Enablement Resources

How can you implement this?

- You can use automation tools such as AWS CloudFormation templates or leverage a Technology Partner solution to provision, secure, connect, and run solution infrastructure. You can use 3rd party tools for Configuration Management, Container Services, and CI/CD. For DevOps components, you can leverage specific AWS services (AWS CodePipeline, AWS CodeBuild, AWS CodeDeploy, AWS CodeStar, AWS CodeCommit, Amazon Elastic Container Service, AWSOpsWorks, AWS Elastic Beanstalk).
- You can explore AWS Cloud Development Kit (AWS CDK) to model and provision your cloud application resources using familiar programming languages.

Additional Resources

- 8 best practices when automating your deployments with AWS CloudFormation
- Best practices for developing and deploying cloud infrastructure with the AWS CDK

Good Example Response

- The team utilized AWS Cloud Development Toolkit (CDK) to create CloudFormation Stacks. They have a separation of four environments from source, build, test, and production. When changes are made to CloudFormation stacks, they undergo a multi-tier test suite which performs a variety of functional, integration, and load tests. After passing all benchmarks in the test environment, final deployment to production occurs. This happens in a staggered manner which performs integration and canary testing continuously until complete deployment. Should errors be found at any stage, a rollback occurs, the team is alerted, and their dashboard is updated.
- Provide an attachment of a file or PDF of the example template

Unacceptable/Incomplete Response

- Lack of concrete evidence for automation in deployment
- A generic description without specifics or examples:
    - We use Terraform and CloudFormation for Infrastructure-as-Code for deploying all our management components.
    - As a code for deploying management components, we leverage the power of Terraform and CloudFormation for Infrastructure. Moreover, we also follow the guidelines of DevOps for all prod and UAT deployments on AWS.
- Partner responding "N/A" or an answer similar to any of the following: "The whole operational activity was provided by the customer team. All process documentation and automation scripts were studied and developed inside customer infrastructure.", "We did not perform this as part of this engagement, this was the customer's responsibility", etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

REL-002

Requirement

Plan for Disaster Recovery and recommend RTO and RPO:

Incorporate resilience discussion and advise an RTO and RPO target when engaging with customer. Customer acceptance and adoption on RTO/RPO is not required.

- Establish a process to establish workload resilience including:
    - RTO and RPO target
    - Explanation of the recovery process for the core components of the architecture
    - Customer awareness and communication on this topic

Criteria for Passing

- Description or documents on workload resilience guidance met the three criteria defined above.
- Description of how resilience is implementation in one (1) of the submitted customer examples including reasons for exception when RTO and RPO is not defined.

Why is this important?

- Recommending an appropriate RTO and RPO based on customer business needs is critical to their disaster recovery plan. For the given workload, you must understand the impact of downtime and lost data on your customer business. There is often some balance required between having a resilient/reliable infrastructure and keeping costs manageable, you should finalize these goals with your customers for the specific workload while being intentional about these trade-offs.

Technical Enablement Resources

How can you implement this?

- Work with customers to define resiliency goals and objectives. Incorporate customer's resiliency needs into on-going project discussion, educate customers on the trade-off between resiliency and cost by working from their business needs.
- Test how each component of the workload fails, and validate the recovery procedures. You can use automation to simulate different failures or to recreate scenarios that led to failures. This approach exposes failure pathways that you can test and fix *before* a real failure scenario occurs, thus reducing risk.
- Consider utilizing AWS Resilience Hub to improve the resiliency of your applications on AWS and reduce the recovery time in the event of application outages.

Additional Resources

- Shared Responsibility Model for Resiliency
- Plan for Disaster Recovery
- Establishing RPO and RTO Targets for Cloud Applications

Good Example Response

- An attached SOP or documentation on workload resilience guidance that meets the three criteria detailed in the requirements above.
- The designed system has an RTO of 5 minutes in the event of an AZ failure. This is accomplished through using a warm-standby approach. In the event of a regional failure, our RTO is 2 hours and is handled through deploying the solution in another region from our CloudFormation Templates, and restoring data from backups. Our

RPO is 10 seconds for our Amazon Aurora Global Database, which has multi-AZ and multi-region deployments for a realized RPO of 1 second.

Unacceptable/Incomplete Response

- Lack of a defined RTO or RPO strategy
- Partner responding “N/A” or an answer similar to any of the following: “The whole operational activity was provided by the customer team.”, “We did not perform this as part of this engagement, this was the customer’s responsibility”, etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

COST-001

Requirement

Develop total cost of ownership analysis or cost modelling:

Determine solution costs using right sizing and right pricing for both technical and business justification.

Conducted TCO analysis or other form of cost modelling to provide the customer with an understanding of the ongoing costs including all the following 3 areas:

- Description of the inputs used to estimate the cost of the solution
- Summary of the estimates or cost model provided to the customer before implementation
- Business value analysis or value stream mapping of AWS solution

Criteria for Passing

- Description of how to develop cost analysis or modeling with the critical components defined above
- Cost analysis example in one (1) of the submitted customer examples. Acceptable evidence types are: price calculator link, reports or presentations on business values analysis

Why is this important?

Understanding the total cost of ownership (TCO) and implementing cost modeling for cloud infrastructure is crucial for customers to make informed decisions and optimize their cloud investments. By gaining insights into the TCO of cloud services, organizations can better allocate their expenses, maximize return on investment, and drive long-term financial success.

Technical Enablement Resources

How can you implement this?

- Use a tool like AWS Pricing Calculator to create an estimate on the costs of using AWS Cloud. Dive deep with your customer, as described in this TCO tools guide, to ensure that they are following AWS Best Practices such as right-sizing their instances, choosing the right pricing models, and using modern technologies/architectures when possible. Your report should attempt to realistically capture their workload as much as possible, including capturing inbound and outbound data transfer rates.

Good Example Response

- Our customer was migrating their containerized application to AWS using a lift and shift method. We provided them a TCO using AWS Pricing Calculator for EC2 and ECS. We used their historical traffic data to demonstrate network costs and instance runtime costs. We also made an effort to right-size their infrastructure, and added in cyclical auto-scaling to demonstrate a variability in instances which would mimic their historical data. Our analysis demonstrated potential cost savings as they had a more stable presence on the cloud, and potential savings as they modernize their application. Our final report given to can be found on the following AWS Pricing Calculator link: https://calculator.aws/#/123456789

Unacceptable/Incomplete Response

- A simple confirmation that this was performed, simply listing some of the components considered in the cost estimate, or not providing a cost analysis example/report (link to pricing calculator, report, presentation, spreadsheet, etc).
- Partner responding “N/A” or an answer similar to any of the following: “The whole operational activity was provided by the customer team.”, “We did not perform this as part of this engagement, this was the customer’s responsibility”, etc. **While we understand not every engagement may require this, partners must have these policies in place. These policies must be followed for at least one (1) of the case studies submitted.**

Resources

Visit AWS Competency, Service Delivery Program Guide to get an overview of the program.

Explore AWS Competency, Service Delivery Program Benefits Guide to understand partner benefits.

Visit How to build a microsite to understand on building a Practice/solution page

Check out How to build an architecture diagram to build an architecture diagrams.

Learn about Well Architected Framework on Well Architected Website

Notices