Application Design and Migration Strategy Template

Client Name

*Location*

*Date*

Table of Contents

[Context and Problem 2](#_Toc74674840)

[Background 2](#_Toc74674841)

[Motivation 2](#_Toc74674842)

[Scope 2](#_Toc74674843)

[In Scope 2](#_Toc74674844)

[Out of Scope 2](#_Toc74674845)

[Requirements 2](#_Toc74674846)

[Functional 2](#_Toc74674847)

[Non-functional 2](#_Toc74674848)

[AWS Architecture Design 3](#_Toc74674849)

[Current Application Documentation 3](#_Toc74674850)

[General Information 4](#_Toc74674851)

[Diagram 4](#_Toc74674852)

[Architectural Decisions 5](#_Toc74674853)

[Architectural Components 6](#_Toc74674854)

[Technical Operations Components 7](#_Toc74674855)

[Software Development Life Cycle (SLDC) Components 8](#_Toc74674856)

[SLDC Environments 9](#_Toc74674857)

[Mandatory Tags 9](#_Toc74674858)

[AWS Annual Run Rate Estimate 10](#_Toc74674859)

[Migration Strategy 10](#_Toc74674860)

[Business Drivers 11](#_Toc74674861)

[Strategy Definitions 11](#_Toc74674862)

[Disposition Tree 12](#_Toc74674863)

[Strategy 13](#_Toc74674864)

[Patterns & Tools 14](#_Toc74674865)

[Testing Considerations 14](#_Toc74674866)

[Cut-over Considerations 14](#_Toc74674867)

[Operational Considerations 14](#_Toc74674868)

[Risks, Assumptions, Issues, and Dependencies 14](#_Toc74674869)

[Open Questions / Parking Lot 15](#_Toc74674870)

[Appendix N 15](#_Toc74674871)

# General Information

|  |  |
| --- | --- |
| Item | Description |
| **Application Name** |  |
| **Application ID** | Any ID this is application is known by in the customer environment |
| **SDLC Environment** | intended target region(s) for this application |
| **Region(s)** | intended target region(s) for this application |
| **# Of Availability Zones** |  |
| **AWS Account #** |  |
| **Business or Technical Function** | describe primary function for this application and any other secondary use case relevant to this design |
| **Primary Contacts** | * Application Owner * Application Architect * Enterprise Architect |
| **Data Classification** | Public | Non-Public | Confidential | Highly Confidential |
| **Service Characteristics** | RPO | RTO | Uptime | Internal Facing | External Facing |
| **Migration Strategy** | Rehost / Relocate / Replatform / Refactor or Rearchitect |
| **Migration Pattern** | Fresh install, synchronous replication, snapshot import etc. |
| **Cutover target date** | dd-mm-yyyy |

Table 2 - General information

# Context and Problem

## Background

* Provide brief but relevant information so that readers have some context as to where and why the proposed changes will be made (which will be described later).
* Assume readers are not familiar with the systems.
* Define any needed terminology.

*During discussion: Do we all understand the current state of things?*

## Motivation

* Why have we decided to make the proposed changes?
* What are the benefits?
* Will there be any downsides to the changes? Why do we think the benefits outweigh the downsides?

*During discussion: Do we agree that there is strong motivation for making a change? Is there something else we have not considered?*

## Scope

Will the solution to this problem apply to other applications/services in the portfolio? How much are we willing to generalize the solution? Declare the limits in the scope and out of scope sections.

### In Scope

* List topics, processes, or systems that are in scope of this design

### Out of Scope

* List anything that may be related to the design but that will not be discussed, and why.

*During discussion: Do we agree that we will only stick to the in-scope topics? If not, then now is the time to bring those topics up and provide reasons for why they are relevant. For example, should this solution apply to more problems? After this please avoid bringing up other topics.*

## Requirements

Document, at a high-level, the functional and non-functional requirements for this architecture.

### Functional

* xyz

### Non-functional

* xyz

# AWS Architecture Design

This is where you describe the AWS architecture design for the application. This part can be quite long and it’s important to keep it clear and organized. The kind of organization will depend on what you’re building. Unless necessary, try to keep one application per document.

Here are some tips on organization:

* If you are making changes across multiple systems, then make a secondary heading for each system and then describe system specific changes in that section. Where possible, try to use separate documents.
* Move any supporting material to the Appendices section.
* If you are considering multiple approaches and technologies then document them in the Architectural Decisions table.
* If the approach is to address the problem in multiple phases/steps, then clearly define those in the Migration Strategy section.
* Make sure to use appropriate headings. This builds an easy to navigate outline in Word and makes it easier to find the right section at a glance.

*During discussion:*

1. *Does the design make sense? Will you be able to provide support for this feature once it’s implemented and you’re on-call? Or will you be able to make changes to it?*
2. *Is this the right design? Are there other options that are better? Are there any important downsides that make this solution not viable? Are the new APIs necessary? Do they have the right structure? Same for new data stores. Etc.*
3. *Is this architecture well-architected? (Use the AWS well architected framework)*
4. *Is this architecture secure? Ensure security/compliance/regulatory requirements are met*

## 

## Current Application Documentation

List existing documentation for the current (on-premises / co-location / other cloud) implementation of this application

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Document Type | Title | Links |
|  | Design/diagram/other |  |  |
|  | Infrastructure inventory |  |  |
|  | Performance reports |  |  |

Table 2 - Current architecture documents

## Architectural Decisions

When more than one option for a given architecture has been considered, document here the options that were analyzed and the rational for selecting one of them.

|  |  |  |  |
| --- | --- | --- | --- |
| Reference | Architectural Pillar | Options | Rationale |
| AD001 | Security | * Option A (selected) – describe option * Option B – describe option * Option C – describe option | Describe motivation for selected option |
|  |  | Add rows as necessary | Add rows as necessary |

Table - Architectural Decisions

## To-Be Architecture Diagram

Use the provided drawio template (can be found in the same location from where this document was obtained) or provide your own. Ensure the diagram is clear and emphasizes the architectural decisions made. It is highly recommended to revisit the base template and adapt it to the specific base configuration of the Landing Zone.

Use Reference numbers in the diagram in order to link them to the Architectural Components details in the next section.

## 

## Architectural Components

Describe each architectural component. As a minimum, ensure all components in the diagram are described. The table below includes sample components.

|  |  |  |  |
| --- | --- | --- | --- |
| **Reference Number** | **Logical Component Name** | **Description** | **Details** |
| **1** | **Internal Active Directory** | Internal Authentication and DNS service | Domain will remain on-premises and will be transitioned to AWS at a later stage |
| **2** | **External Active Directory** | External Authentication and DNS service | Domain will remain on-premises and will be transitioned to AWS at a later stage |
| **3** | **Virtual Machine - ServerA** | File Server | Shared Service used by multiple applications. Will remain on-premises and migrated at a later stage. |
| **4** | **Virtual Machine - ServerB** | File Server | Shared Service used by multiple applications. Will remain on-premises and migrated at a later stage. |
| **5** | **External access** | Access from external customers | DNS servers house public and Internal DNS zones that include records for the following site:   * api.acme.com   + Public Routable address   + Internally Routable address |
| **6** | **Route53 (DNS)** | DNS Services in AWS | * Domains * Usage |
| **7** | **KMS** | EBS Volume encryption |  |
| **8** | **CloudTrail** | Logging and Monitoring | AWS Account activity monitoring |
| **9 / 11** | **Public Subnets** | Public subnet used for web services | Subnet - Subnet ID - AZ1  Subnet - Subnet ID - AZ2 |
| **10 / 12** | **NAT Gateway** | Internet access for private subnets |  |
| **13** | **Direct Connect** | 10 Gbps link to on-premises DC 1 |  |
| **14** | **Transit Gateway** | Provides connectivity across VPCs and On-premises |  |
| **15** | **VPC** | Primary VPC | VPC ID and CIDR Range |
| **16 / 17** | **Private Subnets** | Private subnet used for backend servers | Subnet - Subnet ID - AZ1  Subnet - Subnet ID - AZ2 |
| **18 / 19** | **EC2 Database Instances** | Database | The primary database is deployed on an Amazon EC2 instance in one Availability Zone and a secondary database is deployed on an Amazon EC2 instance in a second Availability Zone. The databases require configuration to enable the following:   * A highly available architecture that spans two Availability Zones * Async replication * KMS to encrypt all root and data volumes   Databased hosted on these instances:   * Instance 1 * Instance 2 |
| **20** | **Load Balancer** | Application Load Balancer | The Load Balancer handles sticky sessions, ensuring that a single user's requests are always processed by the same backend web server. The Load Balancer works in conjunction with Auto Scaling to ensure that web requests are routed to the correct backend EC2 instance based on various factors, such as instance availability and load. The Application Load Balancer routes traffic to the auto-scaling groups.  A Security Group is used to limit access to the ALB.   * Security Group Name: |
| **21 / 22** | **Auto Scaling Groups** | Autoscaling groups are used to scale up and down according to demand | Example: Auto Scaling provides compute resources for web services. Auto-Scaling ensures that the application can scale to accommodate various levels of application load. Additionally, Auto-Scaling is used to ensure that at least 1 instance is available at all times. The Auto Scaling group will automatically replace any instances which are terminated due to failures.  Auto Scaling Group 1 - Name - characteristics  A Security Group is used to limit access to the App Instances in the auto-scaling group.   * Security Group Name:   Auto Scaling Group 2 - Name - characteristics  A Security Group is used to limit access to the App Instances in the auto-scaling group.   * Security Group Name: |
| **22** | **ALB TLS Certificate** | TLS Certificate used to terminate connections on the ALB | AWS Certificate Manager is used to provision, deploy, and rotate a public TLC Certificate on the ALB. A separate wildcard TLS Certificate is used for each environment.   * Prod: \*.acme.com * Test: \*.test.acme.com * Dev: \*.dev.acme.com * Sandbox: \*.sbox.acme.com |
| **23 / 24** | **EC2 instances** | Backend services | Instance names |
| **25** | **AWS Region** | Region for this application | us-east-1 |

Table - Architectural Components

## Technical Operations Components

Describe technical operations components. There can be, for example, core infrastructure shared services.

|  |  |  |
| --- | --- | --- |
| Logical Name | Description | Details |
| Example: Monitoring, Alerting, Inventory, and Change Control | Example: Monitoring, Alerting, Inventory, and Change Control | Example: AWS Services provide monitoring, alerting, inventory, and change control functionality.   * CloudWatch & CloudWatch Logs * Config & Config Rules |
| Example: OS Patching & Remote Access | Example: Patching & Remove Access | Example: AWS Systems Manager (SSM) is used to provide patching and remote access to all EC2 instances. |
| Example: Vulnerability Scanner | Example: Vulnerability Scanning | Example: Amazon Inspector is used to assess applications for exposure, vulnerabilities, and deviations from best practices. After performing an assessment, Amazon Inspector produces a detailed list of security findings prioritized by level of severity. These findings can be reviewed directly or as part of detailed assessment reports which are available via the Amazon Inspector console or API. |
| Example: Security Audit Logging | Example: Security Audit Logging | Example: AWS Services provide additional security functionality.   * CloudTrail provides Security Audit Logging * KMS uses FIPS 140-2 validated hardware security modules to protect encryption keys used for volume and object encryption * Secrets Manager supports the rotation, management, and retrieval of database credentials, API keys, and other secrets throughout their lifecycle |

Table - Technical Operations Components

## 

## Security Groups

Describe Security Groups

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name/ID | Source/Destination | Protocol | Port Range | Description |
|  |  |  |  |  |
|  |  |  |  |  |

Table 4 - Security Groups

## Software Development Life Cycle (SLDC) Components

Describe SLDC Components

|  |  |
| --- | --- |
| Resource | Link |
| Example: Jira Project | Example: Insert link to Jira Project used to manage this application. |
| Example: Source Repository | Example: Insert link to source control repository for this application. |
| Example: CI/CD Pipeline Services | Example: AWS Services are used to store, build, and deploy infrastructure and application components through Sandbox, Dev, Test, and Prod environments.   * CodeCommit * CodeBuild * CodeDeploy * CodePipeline * Cloudformation |

Table - SLDC Components

## SLDC Environments

Describe application environments

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Landing Zone Account | Availability | Purpose |
| Sandbox | Sandbox | 5x12 | Infrastructure as code and application automation activities. |
| Dev | Dev | 5x12 | Application development. |
| Patch | Sandbox | 5x12 | COTS and Application patch application and testing. |
| QA | Test | 5x12 | Quality Control and Integration testing. |
| Prod | Prod | 24x7 | Production operations. |
| Training | Prod | As Needed | Training operations. |

## Mandatory Tags

Describe tags for this application or include a link to where these are defined.

|  |  |  |
| --- | --- | --- |
| Key | Value | Description |
| Example: Business Unit | <business unit name> |  |
| Example: Environment | <environment> |  |
| Example: Product | <product name> | Software associated to primary function of component |

Table - Mandatory Tags

## AWS Annual Run Rate Estimate

Produce an estimate ARR for the design. Can include more than one option for comparison. Use tooling, such as MPA or Migration Evaluator, to produce the estimate.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Annual Total Run Rate** | **One Time Fee** | **Monthly Fee** | **Description** | **Links** |
| **0.1** | $ | $ | $ | Initial estimate |  |
| **Example: This estimate is based on the following assumptions** | | 1. One Year Partial Upfront Reserved Instances (RIs) are used for all Production Instances 2. Daily Change for EBS Snapshots is 0.1% 3. Utilization for all Production instances (EC2/EBS) is 100% 4. Utilization for all Non-Prod instances is 5 days/week at 12hrs/day 5. Does not include Oracle Licensing 6. Includes Sandbox, Dev, Patch, QA, and Prod environments 7. Includes Windows OS licensing for all Windows instances 8. Includes RHEL Licensing for all Linux instances 9. All internal networking and security is provided by Security Groups 10. Pricing is for AWS us-east-1 Region 11. Includes HA and in Region DR capability | | | |

Table - Estimated AWS Annual Run Rate

# Migration Strategy

This is where you describe the ‘R’ Strategy chosen for this application and the associated Patterns. The R Strategy is the high-level approach (i.e., one of the 7 Rs) whilst the Pattern is the technique used to deliver the strategy. For example, an R Strategy of Refactor can utilize a Pattern of automated code refactoring with a given product or solution. Overall, this part can vary in complexity depending the path taken and It’s important to keep it clear and organized.

Here are some tips on organization:

* Review the standard AWS R Strategy definitions (included in this document) and make adjustments based on customer circumstances. Make sure there is a common language that AWS, Customers and Partners understands and agree to.
* Review and update the 7R disposition tree. Include elements that are key to customer outcomes and decision-making process. The decision tree is to be used as a guide to help the customer understand the best path for an application. The outcome of an application passing through the tree can be overwritten. However, that is a good indication that tree might need to be updated.
* Move any supporting material to the Appendix section.
* If you are considering multiple approaches and technologies then document them in the Strategy Decisions table.
* If the approach is to address the problem in multiple phases/steps, then clearly define those in the Strategy section.
* Make sure to use appropriate headings. This builds an easy to navigate outline in Word and makes it easier to find the right section at a glance.

*During discussion:*

1. *Does the strategy make sense? Does it align to key business drivers? Will you be able to provide support for this strategy and patterns once it’s implemented during a cut-over or migration party?*
2. *Is this the right strategy/pattern? Are there other options that are better? Are there any important downsides that make this solution not viable?*
3. *Has this approach been used before?*
4. *Is this pattern secure? Ensure security/compliance/regulatory requirements are met when implementing the pattern and/or using specific tooling.*

## Strategy Definitions

Review/Update based on agreed common language

|  |  |
| --- | --- |
| R Strategy | Description |
| Retire | Decommission the application without migrating or modernizing. It also applies to services that will be replaced by an entirely new solution. |
| Retain | Do nothing and keep running the application in the current location. It also applies to patterns where the application is 'extended' by adding new components in the cloud. |
| Rehost | Rapid migration of servers and applications without architectural, technology or functionality changes. |
| Relocate | Rapid migration of servers and applications to vmware cloud on AWS. |
| Replatform | Enhanced modernization or upgrade of the application/service underlaying components such as OS and Databases. |
| Repurchase | Purchase, configure or customize a COTS (Commercial Of The Shelf) or SaaS (Software as a Service) product. |
| Refactor | Modernization of the application by applying changes to the code base in order to support a modernization pattern and/or changing its architecture (e.g., containerization, serverless) |

Table - R Strategies

## Strategy

Document the Migration Strategy chosen for this application. Include the rationale

## Patterns & Tools

Document the techniques and tools available for delivering a given Strategy. If there is more than one option available document them including the rationale for the preferred choice.

|  |  |  |
| --- | --- | --- |
| Pattern/Tool | Description | Link |
| Example: CloudEndure (preferred) | <Application components in scope of this pattern/tool> | <APG pattern guide link> |
| Example: Rebuild | <Application components in scope of this pattern/tool> |  |
|  |  |  |

Table - Patterns and Tools

## Testing Considerations

How will you ensure your changes are successful?

* You can break up the content here into a similar structure to your design implementation such as per phase or per system.
* Include what manual tests will be performed.

## Cut-over Considerations

Document here any factors that need to be taken in consideration when planning for migration.

* Cut-over dates to avoid
* Skills required

## Operational Considerations

How will you ensure you are ready to operate this application in AWS?

* What kind of issues do we expect to see come up as a result of this change?

# Risks, Assumptions, Issues, and Dependencies

* What could go wrong and what will its impact be? How can we mitigate those risks?
* Is there any risk of data corruption?
* Are there security risks?
* Are there performance risks?
* What assumptions have we made?
* What dependencies do we have to implement this architecture and migration strategy?
* What Issues do we know about in the current implementation?
* What issues do we expect as a result of implementing this architecture?

|  |  |  |  |
| --- | --- | --- | --- |
| Type (R, A, I, D) | Description | Owner | Status |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table - RAID Log

# Open Questions / Parking Lot

List any part of the design that we have not been able to figure out. Could be requirements or risks we don’t know how to handle or parts of other systems we’re not familiar with that can impact our design. The goal here is to acknowledge these gaps and that they will be looked into later, or possibly during the discussion if the group has ideas.

*During discussion: Is it okay to proceed with the design given that we don’t know how some things?*

|  |  |
| --- | --- |
| Item | Description |
|  |  |
|  |  |
|  |  |

Table - Open Questions / Parking Lot

# Appendix N

In general, all content should belong in a designated section.

If some content is not super relevant but may be useful to reference if desired, like a table or graph, then it can be included here. This is especially true for lengthy extra content.