APPENDIX C

Templates and Worksheets

This appendix provides templates you can use as a guideline for producing your own solution design and threat modeling documents, as well as an example Failure Modes and Effects Analysis worksheet.

Solution Design Document Template

Solution design is a process that helps you think through and visualize a solution before you build it. You begin by identifying and analyzing the requirements, then put together a technical solution to be implemented by the development team. The outcome of the solution design process is a document that captures all the details needed to bring clarity to the architecture and design and shows business stakeholders what the solution will look like. This section walks through the different sections that are typically included in a solution design document, with a brief explanation of the contents of each.



For more information on the importance of solution design and some FAQs, see Sheen's two-part series on the significance of solution design in serverless development.

Overview

The opening section of the solution design document provides a brief outline of its purpose, including contextual details related to the problem the proposed solution addresses. It should be concise, usually containing no more than five lines of text.

Revision History

The revision history is a log that records all the changes to this document. This section is particularly useful if you share the document with other teams or external partners. A table listing the version number, date of change, and a summary of changes is enough to keep track of the evolution of the work.

Requirements

This is an important section of the document, where you provide the details of the context summarized in the Overview section. In this section, you refer to business stakeholder documents, task epics and tickets, business decision records (BDRs), architecture decision records (ADRs), etc.

A good knowledge of the requirements is essential to understand the design details discussed later in the document.

Logical Design

In this section, you provide a nontechnical perspective of the solution at a high level for the benefit of business stakeholders, product managers, and engineering teams. You should use basic diagramming notations and narratives that everyone can understand. For this reason, it is better to avoid cloud service icons. Instead, use domain models, flowcharts, sequence diagrams, user journeys, C4 model diagrams, etc.

The contents of this section will be your central resource while discussing the solution with business teams.

Architecture

In this section, you'll provide one or more solution architecture diagrams or links to their sources. The number of diagrams you add here and their contents depend on the problem space and the level of serverless knowledge in the team. You may use cloud service icons in the architecture diagrams to convey the details, as the audience is mostly technical people.

Design Details

This is a core section where the focus is on the implementation details of the solution. You take the different parts of the architecture and expand on them by adding relevant information about API contracts, service quota limits, data validation and encryption requirements, computation algorithms where necessary, workflow details, caching needs, failure recovery mechanisms, etc. If you are maintaining documents such as API specifications separately, you may also want to add links to those here.

The level of detail in this section depends on the technical fluency of your team. It should take into account the principles of the AWS Well-Architected Framework, best practices, cost and sustainability measures, and organizational standards and guidelines.

Data Modeling

In this section, you focus on the solution's data storage needs and event schemas. With myriad data store options, some up-front thinking on how to handle and store data will be of benefit during implementation. For example:

- For RDBMSs and SQL data stores, capture the data entity relationships, keys, indexes, and other details.
- For NoSQL stores such as DynamoDB, it is important that you identify the data access patterns, primary and sort key patterns, item types, indexes, TTL needs, scaling options, backup mechanisms and schedule, etc. Also, remember that the characteristics of each DynamoDB table can be different.
- For an object store such as S3, focus on basic details such as data folders, object key patterns, lifecycle policies, replication needs, etc.

In addition, as Chapter 3 discussed in detail, it is vital to think up front about the structure of the events in an event-driven architecture. The EventStorming exercise usually feeds into this section and helps to identify the event categories, types, and metadata of each event.

Security and Threat Analysis

Security is not an afterthought, and you must incorporate it from the start of your solution thinking. Depending on the complexity of the solution, you can either add the details here or produce a dedicated security and threat modeling document (a template is provided in the following section of this appendix) and add a link to that document.

Cost Estimation

Based on known factors and assumptions, as a team, perform a cost estimation analysis for the proposed architecture and record the findings here. The aim is not to calculate the exact cost of the solution—which would be impossible to do at this stage, anyway—but to create awareness and identify cost blind spots to be mindful of during implementation and operation. Use the guidance and tools discussed in Chapter 9 for this purpose.

Sustainability Measures

Have a discussion about sustainability with your team and record any identified measures, precautions, etc. Refer to Chapter 10 for an understanding of the relevant patterns and best practices.

Observability and Monitoring

As with security, you should start thinking about observability early during the design phase. For example, the observability requirements you capture here will guide how your team adds structured logs in the code, the types of dashboards to set up, and other measures.

Trade-offs

This section should list all the trade-off decisions and associated justifications you make during the solution design process. Make sure that these don't contradict any existing BDRs and ADRs.

Appendix

The appendix is a placeholder for references, diagram scripts, sample data, and other useful materials.

Security and Threat Modeling Template

Threat modeling is a process by which potential threats, such as structural vulnerabilities or the absence of appropriate safeguards, can be identified and enumerated, and mitigations can be prioritized.

This document template follows the STRIDE methodology, introduced in Chapter 4, which describes six threat categories: spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege.



The following articles by Sheen provide further details on the significance of threat modeling in serverless and the steps involved in the threat modeling process:

- "Why Threat Modelling Should Be Part of Your Serverless Development"
- "A Few Steps to Conduct Threat Modelling in Serverless Applications"

Description

This section sets the context of the document. Here, you summarize the application and its characteristics, describing it in a way that all audiences should be able to understand.

Architecture

In this section, you refer to the Architecture section in the solution design document and provide additional context about the application from a technical standpoint.

Data Flow Diagram

Data flow diagrams (DFDs) are used to depict the flow of information using standard notations for processes, data flows, data stores, and external entities. This helps you identify the trusted and untrusted zones of the application.

Assets

As the name implies, an asset is something of value. These are the main items that an attacker might target, and consequently the things that you'll want to protect. They could also be things an attacker might use as an entry point to gain access to other critical assets. Identifying and describing your assets (name, location, criticality level, etc.) is essential in order to take the necessary protective measures.

Attackers

Attackers can come in various forms: business competitors, data miners, spies, organized criminals, terrorists, bots, thieves, employees, and more. Start by grouping them into two categories, external and internal, and try to identify all the possibilities. For example, external attackers could be partners with access to your system's APIs, and internal attackers could be disgruntled employees or untrustworthy system admins.

Exclusions

In this section, you list the items outside of your threat model and responsibility. For example, the Lambda runtime environment is the responsibility of AWS, not yours; similarly, the devices or systems your partners use are outside your realm of responsibility.

Requirements

The requirements for your threat modeling process will help you identify potential threats and formulate threat mitigation plans. Though there are threats that are common to every system, the requirements differ between industries and business

domains. The types of requirements that feed into threat modeling include business, industry, technology, regulatory and compliance, privacy, etc.

Threats

This is where you will spend most of your time as part of the threat modeling process: identifying threats, assessing their risks, analyzing mitigation plans, and thinking of action plans. For every threat you identify, you associate it with a type based on STRIDE, provide a brief description, specify the attacker, assess the risk, provide a remediation plan, and create an action.

Actions

Usually, you associate the actions with the threats you have identified and listed in the previous section. However, it is still valuable to add them here, with the tickets to track them. For example, if one of the threats you've identified is an attacker stealing the API credentials of your partner, you need a process to invalidate the client credentials. Your action could be to create a runbook and document the steps to invalidate API client credentials.

Residual Risks

Here is where you identify the risks that are still possible even after taking all practical preventive measures. For example, even if you have a web application firewall (WAF) and related mechanisms to prevent DDoS attacks, an attacker may try to impersonate users in small batches at intervals, making it difficult to identify and block their attack.

Appendix

The appendix is a placeholder for references, diagram scripts, sample data, and other useful materials.

Failure Modes and Effects Analysis Worksheet

Failure Modes and Effects Analysis (FMEA) is the process of reviewing the components of your software to identify the ways in which your system could fail (the failure modes), the root causes of these potential failures, and the effects they could have on your application. Each failure mode and its effects can then be analyzed by rating the probability, severity, and detectability, in order to understand the risk level. The risk level can be calculated by multiplying the probability rating by the severity rating and adding the detection rating: $(P * S) + D$.

The FMEA process can help you to identify as many points of failure as possible in your system and allow you to prioritize the implementation of remedial steps, such as automated recovery, automated tests, or metric alarms.

Failure Modes and Effects Analysis Worksheet

An example FMEA worksheet would include the following columns:

Component
: List the parts of your system that could fail.

Failure mode
: Describe the failure or error that could occur.

Cause(s)
: Write out any potential causes of the failure.

Effect(s)
: Describe the effects of the failure on your application.

Probability
: How likely is it this failure will occur? Use the probability rating scale.

Severity
: If the failure did occur, how severe would the effects be? Use the severity rating scale.

Detection
: Could an occurrence of this failure be detected pre-deployment? Use the detection rating scale.

Risk
: Determine the risk level of this failure by calculating (*Probability* * *Severity*) + *Detection*.

You may want to include additional columns describing actions to take, mitigations, etc. The subsequent sections provide examples of how you might assign probability, severity, and detection ratings for each potential failure mode.

Failure Modes and Effects Analysis Probability Rating

For each failure mode, you should rate the probability of it occurring, from extremely unlikely to occur to likely to occur frequently. Table C-1 is an example probability rating scale.

Table C-1. Probability scale

Rating	Definition
1	Extremely unlikely (virtually impossible)
2	Low
3	Occasional
4	Fairly likely (repeated)
5	Frequent (almost inevitable)

Failure Modes and Effects Analysis Severity Rating

For each failure mode, you should rate the severity of the effects should such a failure occur, from no discernible effect to complete failure of your entire system or service. Table C-2 is an example severity rating scale.

Table C-2. Severity scale

Rating	Definition
1	No effect on reliability or quality
2	Very minor
3	Minor/low
4	Critical
5	Complete failure

Failure Modes and Effects Analysis Detection Rating

For each failure mode, you should rate the likelihood of the fault being detected by a test suite before deployment to production, from absolutely certain to impossible. Table C-3 is an example detection rating scale.

Table C-3. Detection scale

Rating	Definition
1	Certain (fault will be caught by tests, pre-deployment)
2	High/almost certain
3	Moderate
4	Low
5	Undetectable