Creative Technology Solutions (CTS)

# **Software Design Template**

Version 1.0

## Table of Contents

[***Software Design Template 1***](#_heading=h.gjdgxs)

[**Table of Contents 2**](#_heading=h.30j0zll)

[**Document Revision History 2**](#_heading=h.1fob9te)

[**Executive Summary 3**](#_heading=h.2et92p0)

[**Requirements 3**](#_heading=h.tyjcwt)

[**Design Constraints 3**](#_heading=h.tyjcwt)

[**Rationale 3**](#_heading=h.3dy6vkm)

## Document Revision History

| Version | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 10/18/1998 | Douglas Robinson | Initial |

## Executive Summary

Important Stuff

## Requirements

Important Stuff

## Design Constraints

Important Stuff

## Rationale

Important Stuff

**Recommendation**

The Server operating system has 3 primary considerations: price, scalability, and software compatibility. I’ve found that Rocky Linux 8 will serve our purpose well. On the topic of price, Rocky Linux is free when compared to similar distros of Linux like the popular RHEL (RHEL is free but only for small networks of less than 16 systems which is likely less than we will use); it also beats Windows and Mac OS which also have a licensing fee. Still, on price, Windows-compatible software to support our server has additional licensing fees associated with it like Microsoft SQL, Windows will easily end up being the most expensive for any tools we may want to use. Mac OS gets the worst of all worlds as we would have to find an extinct iso and would still be burdened with Windows garbage licensing fees for something like Azure for our database. Given how the other two options are much more expensive Linux wins on the price side of things, so what about scalability? Linux has magical tools like Kubernetes and docker that are free (if our company is under 250 employees and has less than 10 million in annual revenue), and allow for near-infinite scaling by just initializing a container and having the software do load balancing for you. Windows Server 2022 does have containerization but like all things Microsoft you’re paying a premium for something free on Linux, and Mac Server OS doesn’t have any inbuilt containerization features. For scalability, it’s a tie between Windows and Linux. Last on the list of things to go over is software compatibility; Linux easily has the most open source and free tools associated with it and will have compatibility with MOST software we want that is free. Windows has equivalent proprietary software to Linux, but they come with two major downsides, price and transparency. Large companies have been known to put backdoors into their software at the behest of nation-state actors and while that concerning on its own what’s even more concerning is there have been incidents where these backdoors were stumbled upon by malicious actors and exploited; It is entirely a black box to the security of all windows proprietary software. Mac Server OS so it is losing compatibility by the day to up-to-date software. Overall I’d say Linux wins this category, and now to tally up our scores we get 3,1,0 with Linux being a clear frontrunner for these categories.

Linux has often been described as having an onion architecture the lowest layer of the OS is the kernel, this layer sits just above the hardware layer and directly manages your computer resources by directing various processes. The next layer up is the shell, this acts as the most direct intermediary for the user and the kernel allowing the user to write abstracted commands to the kernel. Above that are the applications and utilities that further abstract users from the kernel layer by adding another layer on the shell, this allows more complicated and human-intuitive software to be used to interact with the kernel. There are ways for a user to interact more directly with the hardware through compiled machine code, but the kernel holds executive power over whether certain commands will work.

Storage management is very important for us as we will need to track are large player base and active games, to accomplish this we will consider 2 factors price and size, A cloud-based solution like AWS would seem like a good choice at first glance, however, the price would be relatively high for just being a glorified string repository; What I would suggest instead is something like SOLR and MariaDB for our storage management needs. I picked Solr because it is a free indexing software that can give us an easy way to search through the thousands of players and games. MariaDB was picked because it is free and tried and tested, it is one of the most highly-rated DBs of all time. Another benefit is that we can make containers running MariaDB Galera Cluster on our already existing infrastructure and turn our game servers into a mini DB cloud.

Linux implements several techniques to get better memory management; One such feature is swapping. swapping is a very intuitive and simple way of getting back more memory by putting the memory (page) that hasn’t been used recently back into swap space. Linux has segmentation, segmentation works by well segmenting the pages of memory splitting the data into segments, and then connecting them with other relevant segments. Compaction is another technique in Linux and it works by doing an overhaul of the memory that is loaded by grouping all the space and regrouping all the processes back together from other forms of memory management.

The next part of distributed networks and systems has been touched on a couple of times before but to go into further detail… I think the whole application should be containerized and using Kubernetes we can easily coordinate our humble network of game servers. This type of software is pretty amazing in the sense that if there is an outage in a server it can tell other servers to spin up new containers to compensate for the loss of a few servers. The one major downside is that Kubernetes is not very user-friendly for beginners and is a bit of work to set up initially. MariaDB Galera Cluster works similarly by creating redundant shards across the network and can handle a couple of server losses on its own without human intervention. The best part about all of this is that they are guaranteed to be compatible because of containerization. The only flaw that could arise is if we only use one internet connection for all of our servers and then we’d have a glaring single point of failure.

The last important thing is security. Linux gives admins a variety of tools for managing user permissions and has very strict and robust file access permission controls. It is also compatible with AES, RSA, SHA, and many more encryption methods. Linux also has its configurable system firewall that by default disallows incoming SSH (port 22). Linux also has logs that contain messages from the kernel, mail, and logon attempts. login activity, cron jobs. To put these all to use for our purpose we first need to encrypt our communications to the client and other servers; the next thing that needs to be done is strict access control and only give permissions that users and admins need; after that, we need to lock down the system firewalls on our machine to only allow approved network traffic; and last for the users sake we need to use PKI and get a valid certificate for our website to protect our domain from being poisoned, or at least give warning to the user that it was.