Alexey Kaschenko

Location: Voronezh, Russia
Website | Telegram | GitHub
Email: contact@alexeykaschenko.ru | Phone: +7 980 542-80-07

Backend Developer

I develop in Go, C++, C#/.NET and administer Linux servers, applying DevOps and SRE practices along the way. I am currently a master's student at the Faculty of Computer Sciences of Voronezh State University. In my free time I write services for my own servers and study new server technologies.

Technology Stack

Languages	:	**Go, C++, C# / .NET, Bash, SQL**
Backend	:	**REST API, gRPC, PostgreSQL, MySQL, SQLite, Redis, Nginx, JetStream NATS, RabbitMQ, Kafka**
Infrastructure	:	**git, GitLab, GitLab CI/CD, Linux, Docker, Docker Swarm, Kubernetes, Ansible, Vagrant, Taskfile, Cockpit, Webmin, Prometheus, Grafana, ELK (Elasticsearch, Logstash, Kibana)**
Familiar with	:	**C, Wolfram Mathematica, LaTeX, HTML, CSS, JavaScript**

Education

Voronezh State University — Voronezh, Russia
Bachelor, AMM, Applied Mathematics and Informatics — *2022 – 2026*

Voronezh State University — Voronezh, Russia
Master, FCS, Mathematics and Computer Science — *2026 – Present*

Projects

systemd Service Management System — Go, systemd, gRPC, REST, PostgreSQL
Personal project — Source code

A system for centralized management of systemd services on a group of servers. Every server runs an agent that itself establishes an outgoing gRPC connection to the control node, so the servers need no open ports, and machines behind NAT connect the same way as a VPS with a public IP. The connection is protected by mutual TLS authentication; the control node maintains its own certificate authority and issues and revokes agent certificates. Users, access policies, API tokens and the audit log are stored in PostgreSQL, and permissions are granted down to the server, service and action. Services can be managed through a web panel or the REST API, for example from a Telegram bot. On the servers themselves the agent runs as an unprivileged user and calls systemctl through sudo only from a whitelist.

Invisible Geometry Visualization for Source Engine Maps — C++, SourcePawn, Source Engine
Personal project — Source code

An extension for srcds servers that draws invisible walls, ladders, buttons and other hidden map objects for players. The core is written in C++ and reads the map's BSP file directly, without third-party libraries, including decompressing LZMA-compressed lumps. The shape of each brush is reconstructed from the planes of its faces using the Sutherland-Hodgman algorithm, and degenerate and broken polygons are discarded. The extension decides which brushes to treat as invisible based on surface flags, materials and entity properties. The finished edges are passed to a SourcePawn plugin, which draws the nearest of them as beams. Each map can have its own settings in a separate config.

GameBanana Website Monitoring — Go, NATS JetStream, SQLite
Personal project — Source code

A set of services that tracks new and updated mods on the GameBanana website. The tracker polls selected categories through the site's API, remembers already found mods in SQLite and publishes each event to NATS JetStream. The other services subscribe to these events independently of each other, so a new handler can be added without changing anything in the tracker. A separate notification service sends announcements and errors. As an example event subscriber I implemented the gb-mms service, which downloads the archive, unpacks it together with nested archives, checks the BSP files, sorts the maps into folders, uploads them to FTP and adds them to the servers via RCON.

Command Line Utility for FTP Go, Cobra, FTP

Personal project *Source code*

A CLI for uploading and downloading files and whole folders to and from an FTP server with one command. Servers are described once in the config as profiles, and passwords are stored separately in an owner-only file or passed through an environment variable. Files are transferred in parallel, each worker has its own connection. Failed transfers are retried, and anything that still could not be transferred is printed as a list at the end. Speed can be capped with a common limit for the whole run. All remote paths are resolved relative to the profile's base directory and cannot go beyond it. The utility supports FTPS, a dry run without transferring and skipping already uploaded files.

File Server for Go Go, net/http

Open source library *Source code*

A Go library, an alternative to the standard http.FileServer. It adds a styled file list with sorting, downloading folders as a zip archive with streaming assembly and size limits, hiding folders with a marker file, and a request rate limiter per IP and for the whole server. It protects against escaping the root directory and serves HTML and SVG in a sandbox via CSP. Used to serve static files on my website.

Personal Website Go, htmx, PostgreSQL, Ansible, Nginx

Personal project *alexeykaschenko.ru*

A personal website with a blog and project pages in Russian and English. The backend is written in Go using the standard library, pages are rendered on the server with html/template, and navigation between them works through htmx without full page reloads.

SRCDS Game Server Deployment (IaC Approach) Ansible, Vagrant, Taskfile

Personal project *Source code*

A lab of two Vagrant virtual machines, where one serves as the control node for Ansible and the other runs an srcds game server with MySQL and Nginx. The whole lab is brought up from a clean Ubuntu 22.04 with one command via Taskfile. The configuration is split into Ansible roles for the system, the network, MySQL, Nginx, the server itself and its plugins. The roles configure ufw, key-only SSH login, a systemd unit for the server, logrotate and journald, and Nginx serves the server content. A separate deploy role synchronizes content via rsync, compiles the plugins and reloads the map. The playbooks are idempotent, and individual roles can be applied by tags. Support for another distribution is added with separate variable files without editing the roles themselves.