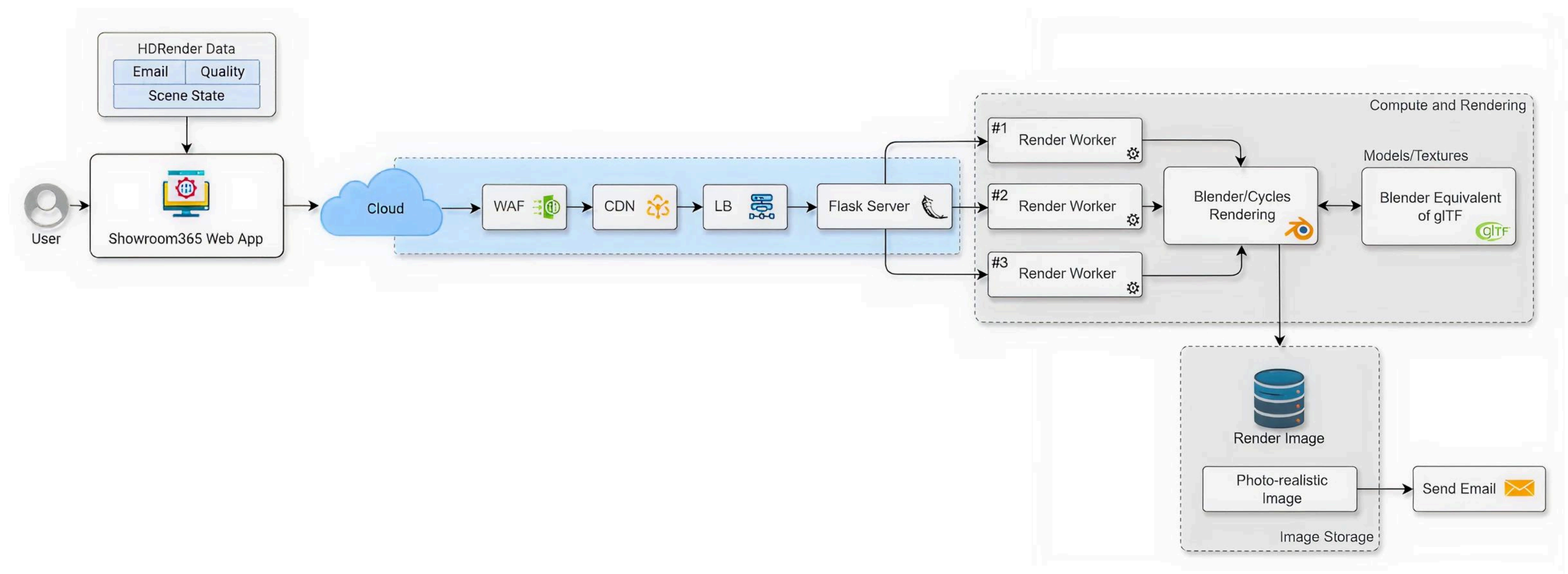


Spaces365 – HDRender Architecture

Executive Summary

HD Render Architecture is a white-label, cloud-native rendering platform that delivers high-quality 1K–4K images and videos without managing GPU infrastructure. Built for scalability, security, and brand-agnostic deployment, it enables enterprises to offer browser-based visualization today, with real-time walkthroughs coming next.

High-Level Architecture Overview Diagram



Key Components & Interactions

HD Render is a cloud-native, API-driven rendering platform that abstracts the full offline rendering pipeline into stateless, containerized GPU workloads.

The client tier exposes REST-based APIs and a browser-based UI for job submission, configuration, and status tracking. Rendering requests include scene metadata, camera parameters, output resolution (1K/2K/4K), render presets, and delivery preferences. All requests are validated and queued asynchronously.

The API layer is implemented using Python (Flask) and serves as the orchestration entry point. It handles:

- Job validation and normalization
- Scene manifest generation
- GPU job dispatch
- Render status polling
- Output delivery triggers

The render orchestration layer converts incoming jobs into isolated container workloads. Each job runs inside a Docker container preloaded with:

- Blender (headless mode)
- Cycles renderer
- Python-based scene assembly and automation scripts

The GPU execution layer runs on RunPod Serverless GPU instances, enabling elastic, on-demand provisioning of GPU compute without persistent node management. Each render job executes independently, ensuring fault isolation and horizontal scalability.

Deployment

HD Render follows a stateless, container-first architecture. All application services and rendering workloads are packaged as Docker images, enabling consistent execution across environments.

GPU workloads are executed using RunPod Serverless, allowing:

- Zero idle GPU cost
- Automatic scale-up based on render queue depth
- Parallel execution of independent rendering jobs

The API layer can be deployed behind any standard reverse proxy and CDN. No persistent GPU nodes, schedulers, or manual scaling policies are required.

Security & Compliance

Each rendering job executes in an isolated container with no shared filesystem or GPU memory access between jobs. This prevents data leakage across tenants in white-label deployments.

Scene assets and outputs are short-lived and access-controlled via signed URLs. API endpoints enforce authentication and request-level validation to prevent unauthorized job execution.

Business Benefits & Alignment

HD Render seamlessly integrates into enterprise workflows, reducing costs, accelerating asset delivery, and enabling scalable, automated visualization across teams.

Strategic Positioning

HD Render is architected as a white-label platform rather than a turnkey rendering tool. Partners can deploy fully branded visualization solutions, customize user experience and workflows, and offer high-resolution static and video rendering directly from their own applications. The core cloud rendering engine, GPU scaling, and orchestration remain centrally managed, ensuring consistent performance, reliability, and future feature expansion, including real-time walkthroughs and AI-driven enhancements.



Roadmap

The current platform supports automated 1K, 2K, and 4K image and video rendering. The next phase introduces real-time walkthrough rendering for interactive visualization. Future capabilities include AI driven scene optimization for automated lighting and realism enhancements, followed by multi-cloud GPU federation to provide global elasticity and redundancy.

References

1. Blender – <https://docs.blender.org/>
2. Cycles Renderer – <https://www.cycles-renderer.org/>
3. glTF (Khronos Group) – <https://www.khronos.org/glTF/>
4. AWS CloudFront (CDN) – <https://aws.amazon.com/cloudfront/>
5. RunPod Serverless GPU Instances – <https://www.runpod.io/serverless-gpu>
6. Docker – <https://www.docker.com/>
7. Python – <https://www.python.org/>
8. Flask – <https://flask.palletsprojects.com/>
9. SMTP (Simple Mail Transfer Protocol) – <https://www.rfc-editor.org/rfc/rfc5321>