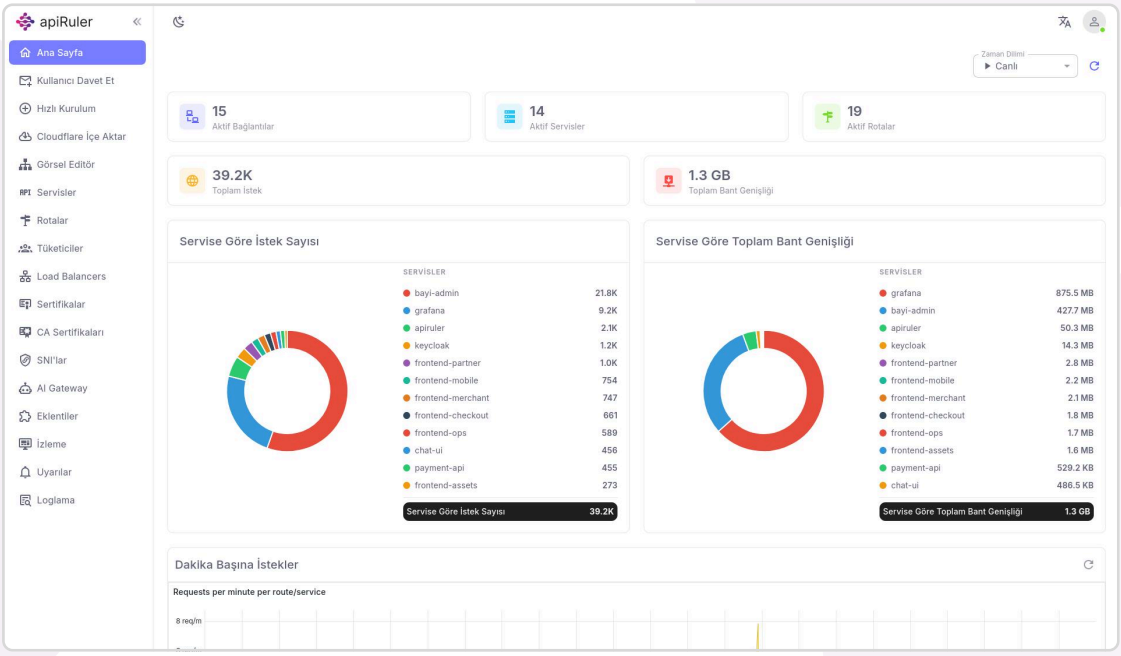
apiRuler

Data Sheet

Cloud

On-Prem

Container



Highlights

API key auth and ACL-based access control at the gateway

Rate limiting and request body validation before traffic hits backends

Edge caching for static assets shared across many frontends

Upstream load balancing with active health checks

Grafana dashboards for metrics, logs, and request origins

Manager UI plus Keycloak SSO for day-to-day operations

API Gateway and Lifecycle Management

Apiruler is an API management platform that puts a unified gateway in front of your services. Built on Kong, it delivers routing, authentication, rate limiting, caching, and observability—with a Manager UI for everyday operations.

Secure every consumer with API keys and ACL groups, validate request bodies at the edge, and throttle abusive traffic before it reaches your backends.

Run the same stack in the cloud, on-prem, or as containers—from a single team service to multi-frontend platforms with dozens of clients behind one gateway.

Highlights

Gateway Security

Using a layered plugin chain at the edge, Apiruler secures APIs before traffic reaches your backends. Key authentication, ACL groups, and request validation stop unauthorized and malformed calls at the gateway. Rate limiting protects high-traffic routes from abuse, while body checks enforce contracts—such as integer-only payment amounts—so bad data never hits application code. Consumers map cleanly to frontends and partners, with blocked groups denied by policy.

Routing, Caching, and Scale

Apiruler routes host-based traffic for many frontends through one gateway —ideal for platforms consumed by dozens of clients. Static assets are edge-cached with proxy-cache so CSS, JS, fonts, and images are served quickly without overloading origin services.
Payment and other backends sit behind a single upstream with round-robin load balancing and active health checks. Unhealthy replicas are pulled from rotation automatically so traffic keeps flowing during failover.

Visibility and Operations

Apiruler metrics flow into Prometheus and Grafana for live dashboards— request rates, upstream health, and gateway performance without a separate monitoring stack.
Request origins resolve client IPs to country and map views via GeoIP and Loki. Day-to-day control runs through the Apiruler Manager UI with Keycloak SSO, so teams manage services, routes, consumers, and plugins from one place.

Features

Gateway & Security

- Host-based routing
- Reverse proxy gateway
- API key authentication
- ACL consumer groups
- Request body validation
- Rate limiting per route/IP
- CORS policy control
- Plugin chain at the edge
- 401/403 policy enforcement
- Consumer-to-frontend mapping
- Blocked group deny lists
- Header-based auth
- Contract checks before backend
- Multi-service route map
- TLS termination (Traefik/LE)
- Pre-function Lua hooks

Scale & Performance

- Upstream load balancing
- Round-robin targets
- Active health checks
- Automatic failover
- Horizontal backend replicas
- Proxy-cache for static assets
- Shared edge cache TTL
- Multi-frontend platforms
- 25+ client scale pattern
- Memory cache strategy
- Cache status headers
- Payment API duplication
- Weight-based targets
- Unhealthy target drain
- High-traffic route throttle
- Asset path caching

Visibility & Ops

- Grafana dashboards
- Prometheus metrics
- Kong /metrics scrape
- Request Origins (GeoIP)
- Country and map views
- Loki log pipeline
- Alloy log shipping
- Manager UI operations
- Keycloak SSO
- Services, routes, consumers
- Plugin management
- Non-destructive deck apply
- CI/CD Kong apply
- Prod and demo profiles
- Cloud / on-prem / container
- Centralized observability
- Upstream health views
- Day-2 operations ready

Specifications

Apiruler Performance Benchmarks

Configuration	Proxy only	+ Rate limit	+ Key Auth	+ Basic Auth
Requests / sec	135,320	113,803	100,305	96,306
P99 latency	5.7 ms	7.84 ms	9.19 ms	9.38 ms
P95 latency	3.55 ms	3.65 ms	4.4 ms	4.67 ms

Benchmarks run on EC2 C5 instance with 16-core cpu 32GB memory.

Resource Sizing Guidelines

Tier	Development	Small	Medium	Large
Entities	< 100	< 1,000	< 10,000	< 50,000+
Latency	< 100 ms	< 20 ms	< 10 ms	< 10 ms
Throughput	< 500 RPS	< 2,500 RPS	< 10,000 RPS	< 10,000 RPS
CPU	1-2 cores	1-2 cores	2-4 cores	8-16 cores
Memory	2-4 GB	2-4 GB	4-8 GB	16-32 GB

Deployment Models

On-Prem

Docker Compose

Bare metal or VM

Full control of environment

Container

Docker, Kubernetes

4 GB min / 8 GB recommended

Gateway as a service