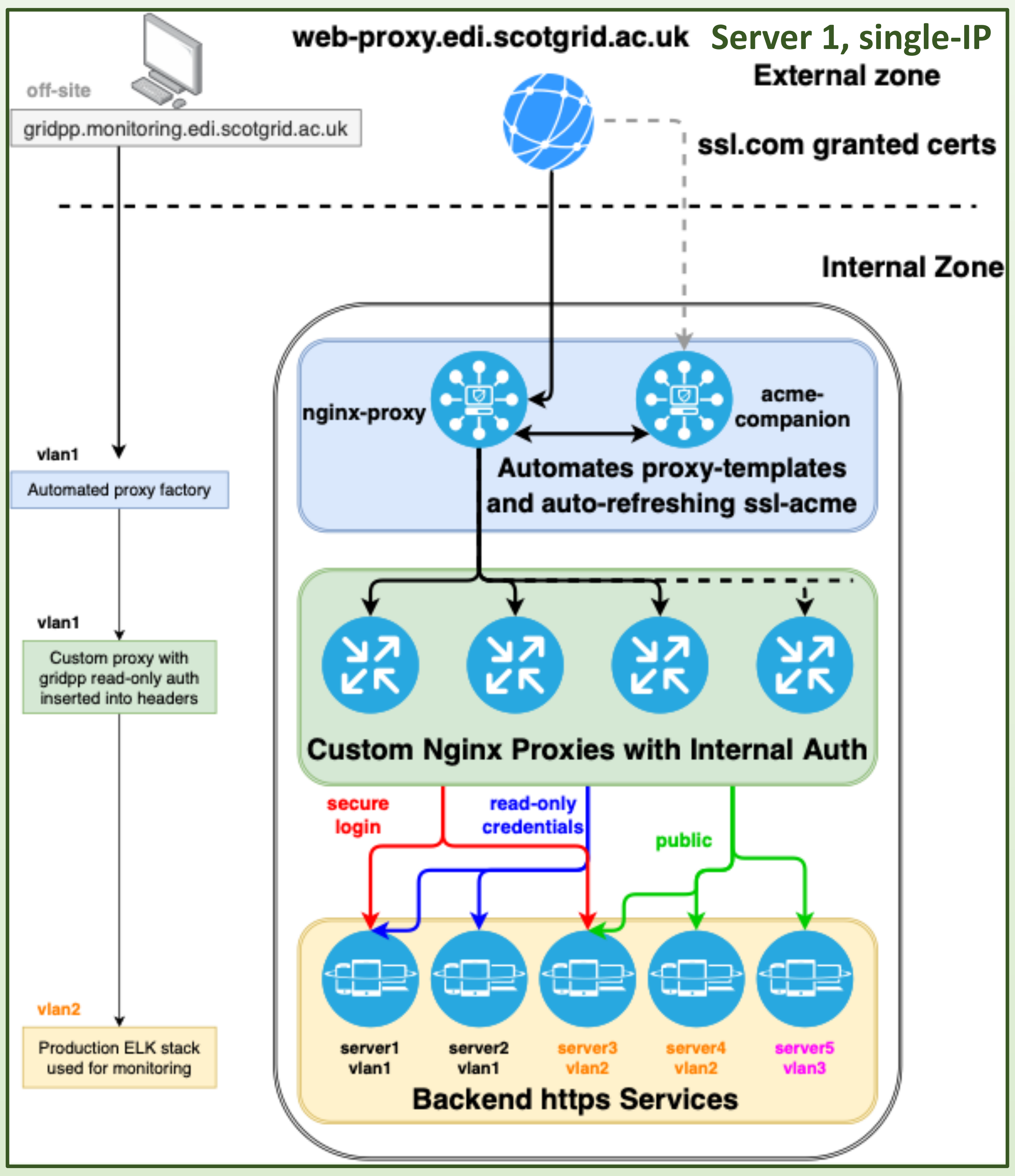
Flexible and Resource-Light Monitoring Platform for a WLCG-Tier2

Robert Currie (**rob.currie@ed.ac.uk**), Wenlong Yuan (**wenlong.yuan@ed.ac.uk**)

THE UNIVERSITY of EDINBURGH

Automated HTTPS Front-end Management



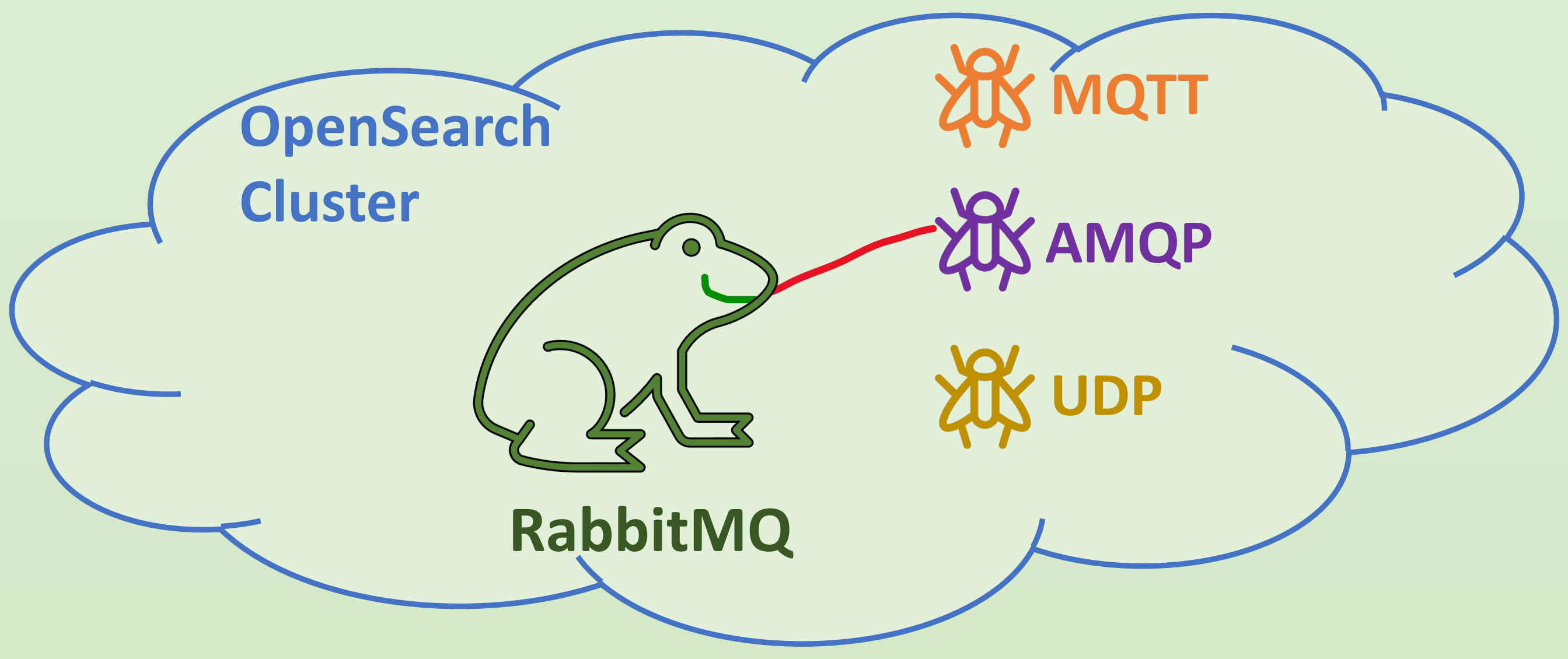
Access to all web services is brokered via a nginx factory.
This allows for automated creation & renewal of SSL certificates for multiple DNS domains.

Controlling a delegated DNS nameserver has allowed for rapid prototyping and development whilst supporting many services within an extremely limited IPv4 allocation.

Nginx Proxies also allow re-writing headers on requests. This allows us to intercept requests and automatically login to provide public services using limited scope functional accounts.

Brokering all https access through a single host allows us to keep repeated common configuration overheads to a minimum.

Containerization allows for individual components to be kept almost stateless with complex inter-container networking handled in simple Docker and Docker-Compose configurations.

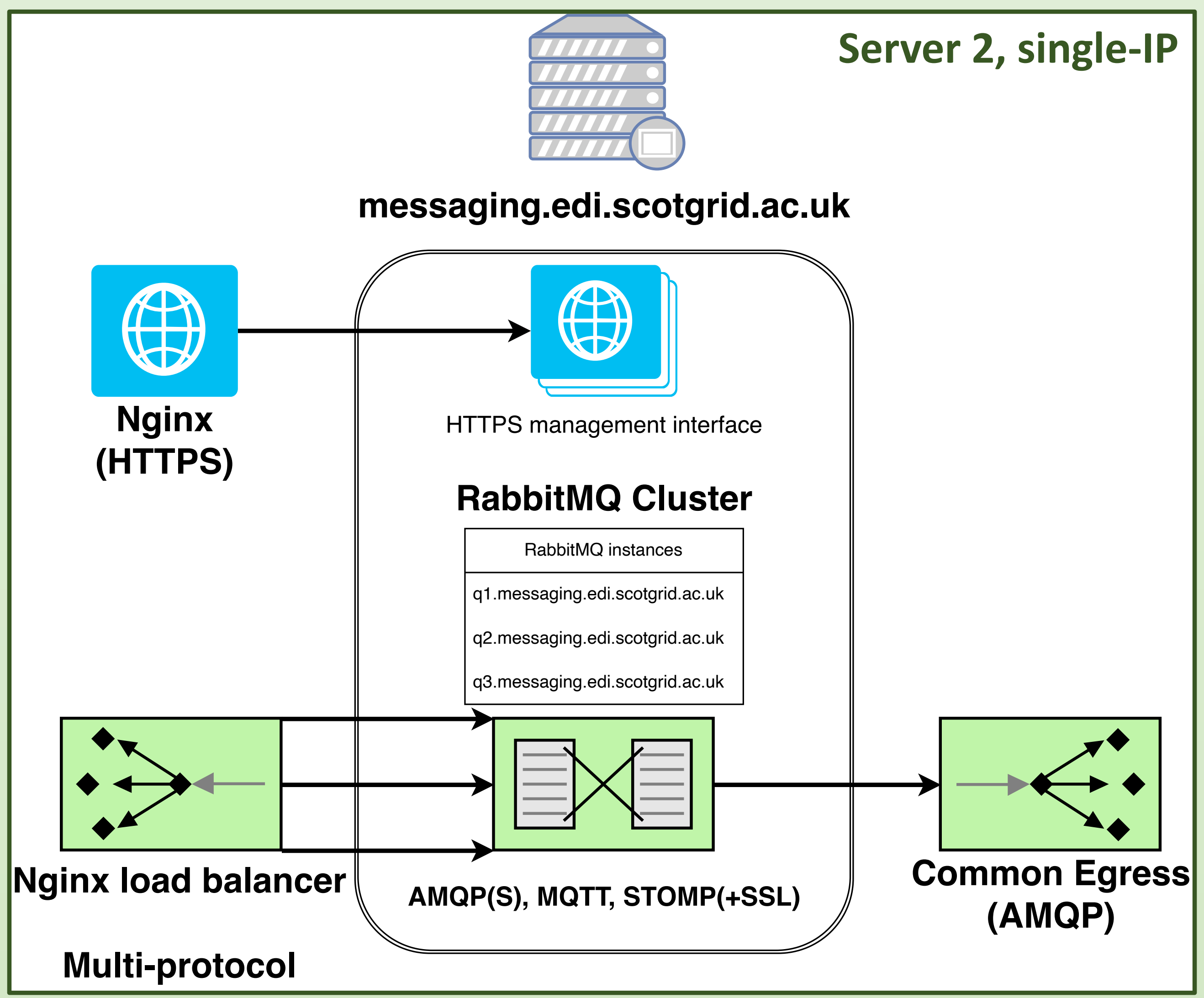


Resilient Multi-Protocol Messaging

Using technologies from our web-proxy, we have also deployed a RabbitMQ stack using stateless containers.

Using auto-managed certificates with containerized services allows for a quorum of RabbitMQ instances to be configured to support redundant message queues.

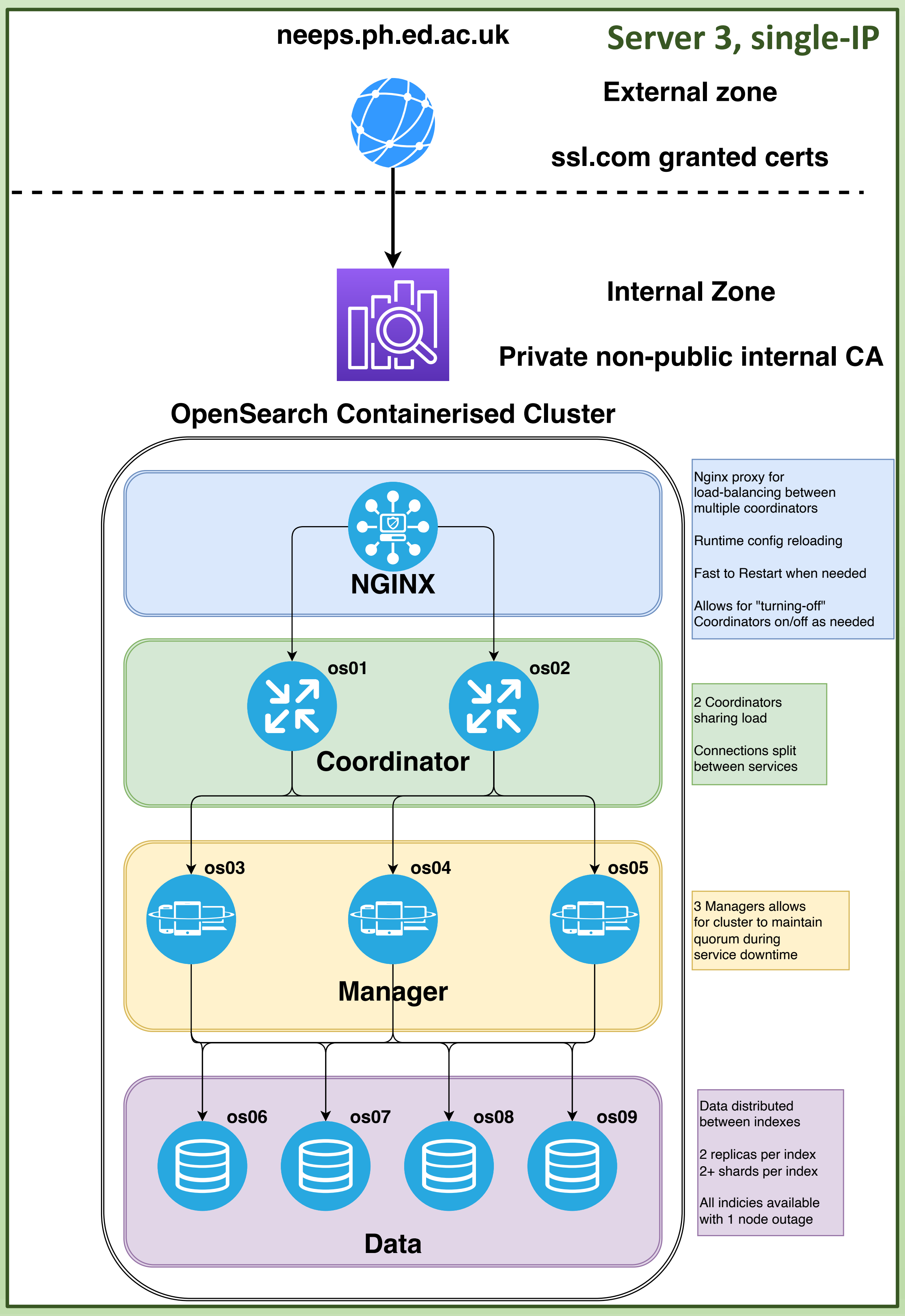
Making use of a dedicated Nginx proxy also allows us to support load balancing across multiple protocols and multiple nodes.


Server 2, single-IP
messaging.edi.scotgrid.ac.uk
Nginx (HTTPS)
HTTPS management interface
RabbitMQ Cluster
RabbitMQ instances
q1.messaging.edi.scotgrid.ac.uk
q2.messaging.edi.scotgrid.ac.uk
q3.messaging.edi.scotgrid.ac.uk
Nginx load balancer
AMQP(S), MQTT, STOMP(+SSL)
Common Egress (AMQP)
Multi-protocol

Supported Production Monitoring Flows:

FluentD (system logs) → OpenSearch
FluentD (security logs) → Buffer → OpenSearch
Custom UDP Metrics → Custom Binary Ingestor → RabbitMQ → LogStash → OpenSearch
MQTT metrics (lab environmental monitoring) → RabbitMQ → LogStash → OpenSearch
Custom STOMP code → RabbitMQ → LogStash → OpenSearch

Resilient OpenSearch Administration



Nginx proxy for load-balancing between multiple coordinators

Runtime config reloading

Fast to Restart when needed

Allows for "turning-off" Coordinators on/off as needed

2 Coordinators sharing load

Connections split between services

3 Managers allows for cluster to maintain quorum during service downtime

Data distributed between indexes

2 replicas per index
2+ shards per index

All indicies available with 1 node outage

Using containers also allows us to build a complex OpenSearch stack, sharing load between multiple components.

Building an OpenSearch cluster of 9 nodes this way allows us to build a resilient cluster with individual components able to be upgraded and restarted on a rolling basis without service outage.

Deploying this service on one dedicated node has been possible due to the use of well-configured ZFS storage.

This has allowed us to re-use old grid storage equipment to provide a backend for a modern storage stack making heavy use of podman and ZFS atop an Alma Linux 9 host.

Summary

Strategies Inspired by Virtual Hosting technologies have allowed us to deploy a resilient monitoring stack for use with our Tier2 site.

This has only been possible through the heavy use of the following key technologies:

- ✓ **Nginx proxies**
- ✓ **Dedicated DNS nameserver**
- ✓ **Containerization technologies** (docker and podman)

Combining these technologies whilst trying to keep all components stateless has allowed us to build a highly-resilient and expandable service stack.

This has all been deployed using minimal amount of hardware (**using limited IPv4 allocation of 3-IPs in 2 vlans**) reducing classical overheads associated with server deployment and maintenance.

This has also been possible due to the re-use of an older grid storage node to provide resources to host an OpenSearch cluster.