BAFAR
Infrastructure Design

Developed by Rayanesh Mobin Hooshyar

Esfand 1403

1. Floating IP Implementation

➢ **Overview:**

- The floating IP is implemented using **Keepalived** software and **VRRP (Virtual Router Redundancy Protocol)** . This setup ensures high availability by directing all incoming traffic to the floating IP address, which acts as a single entry point for clients.

➢ **Failover Mechanism:**

- In the event of a failure or outage on the **master server** , responsibilities are automatically reassigned to an alternative server with a lower priority designation.
- Keepalived continuously monitors the health of the master server and triggers failover if necessary, ensuring minimal downtime.

2. Load Balancing System

➢ **Architecture:**

The load balancing system is positioned behind the floating IP and operates as follows:

1. **Floating IP Layer** : All incoming traffic is directed to the floating IP.
2. **Swarm Networking Layer** : Traffic is managed by Docker Swarm's built-in networking, ensuring seamless communication between containers and services.
3. **Nginx Reverse Proxy Layer** : Within the Swarm cluster, an **Nginx reverse proxy** forwards traffic to the targeted service based on predefined rules.
4. **Targeted Service Layer** : Nginx directs traffic to the appropriate backend services running inside the Swarm cluster.

➢**Key Features:**

- **High Availability** : Automatic failover ensures continuous service availability.
- **Scalability** : Dynamic scaling capabilities adapt to changing traffic demands.
- **Performance Optimization** : Efficient traffic management reduces latency and improves response times

3. MariaDB Cluster

- **Databases: Replication Design**
  - **Replication Type** : Master/Slave with GTID-based asynchronous replication.
  - **Binlog Format** : Row-based binary logging ensures accurate data replication.

- **MaxScale Load Balancer**
  - **Overview** : All database traffic is proxied through **MaxScale** , ensuring optimal distribution of database requests.
  - **Location** : MaxScale resides within the Swarm cluster, maintaining high availability and accessibility from the internal network.

3. MariaDB Cluster

- **Read/Write Split**
  - **Write Queries** : Directed to the master server for data integrity and consistency.
  - **Read Queries** : Dispatched to synchronized slave servers for load balancing and reduced latency.

- **Failover Mechanism**

  - If the master server becomes unresponsive for 90 seconds, MaxScale automatically promotes one of the slave servers to the master role.
  - Remaining slave servers are reconfigured to synchronize with the new master.
  - If the original master recovers, it is reintegrated into the system as a slave.

4. Web Services

➢ Overview

Web services and their dependencies are hosted within the Docker Swarm cluster. Dependencies include:

- **MariaDB** : Interfaced with MaxScale for database access.
- **Redis** : Used for caching purposes.
- **Shared Storage** : Ensures accessible and consistent data storage across services

➢ Shared Storage Configuration

- **GlusterFS** : Configured to manage persistent data effectively.
- The /mnt path is shared among all servers in the Swarm cluster.

➢ Failover Strategy

- In the event of a service outage on a server, affected services are automatically rescheduled and redistributed to other operational servers within the cluster.

➢ Scalability

- Swarm's inherent scalability allows container services to scale dynamically based on demand.

5. Deployment Strategy

➢ CI/CD Implementation

- The Continuous Integration/Continuous Deployment (CI/CD) strategy leverages **Gitlab-CI** and **Ansible** for automated deployments.

➢ Gitlab-CI Workflow

1. **Initial Stages** : Executes 'Test' and 'Build' processes.
2. **Deployment Trigger** : Activates a deployment webhook exclusively for tag events prefixed with production .

➢ Ansible Workflow

- Ansible invokes deployment processes via the webhook triggered by Gitlab-CI.
- An **Nginx server** is configured to return deployment status, ensuring transparency and traceability during the deployment phase

6. Monitoring and Alerting

➢ Overview

- To ensure the health and performance of the infrastructure, we use a robust monitoring stack consisting of **Prometheus**, **Grafana**, and **Alertmanager**. This setup provides real-time insights into system metrics, enables proactive issue detection, and facilitates timely alerts for critical situations.

➢ Monitoring Components

1. **Prometheus**
   - **Role**: Prometheus serves as the primary time-series database and monitoring engine.
   - **Metrics Collected**:
     - **Incoming HTTP Traffic**: Monitors request rates, response times, and error rates for all services.
     - **Host Resources**: Tracks CPU usage, memory consumption, disk I/O, and network bandwidth across all servers.
     - **Container Metrics**: Monitors container-level resource utilization (CPU, memory, disk, and network) via Docker Swarm integration.

6. Monitoring and Alerting

2. **Grafana**
    - **Role**: Grafana is used to visualize Prometheus metrics through customizable dashboards.
    - **Key Dashboards**:
        - **HTTP Traffic Dashboard** : Displays trends in incoming requests, response times, and error rates.
        - **Host Resource Dashboard** : Provides an overview of server health, including CPU, memory, disk, and network utilization.
        - **Container Health Dashboard** : Shows the status and resource usage of all containers in the Swarm cluster.\

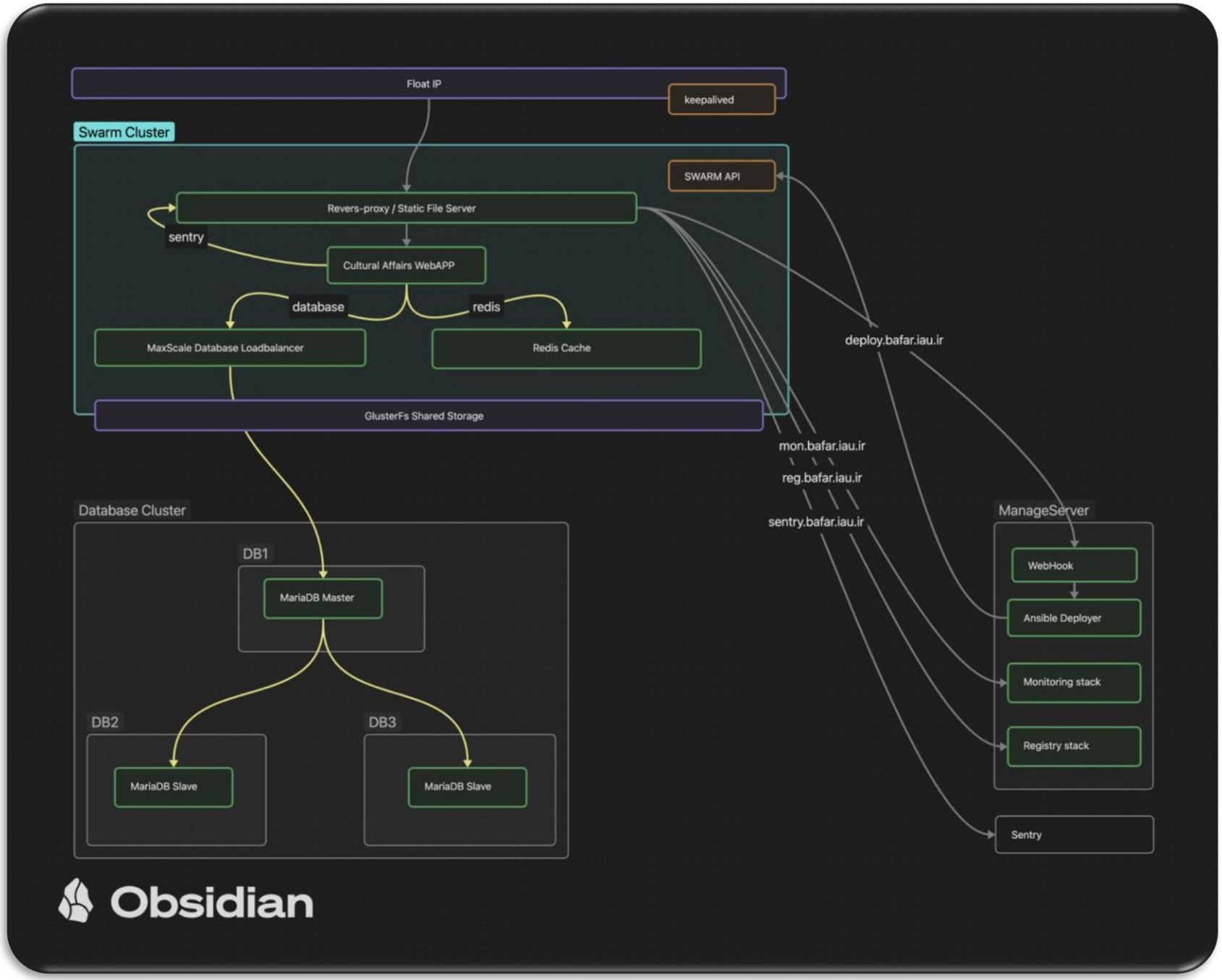
6. Monitoring and Alerting

3. Alertmanager

- **Role**: Alertmanager handles alerts sent by Prometheus and routes them to appropriate notification channels.

- **Alerting Rules :**

  - **Critical Situations :**
    - High CPU or memory usage on any host exceeding predefined thresholds.
    - Disk space utilization above 85%.
    - Network outages or high latency detected in floating IP failover mechanisms.
    - Redis evictions due to insufficient memory.
  - **Service Degradation :**
    - Increased error rates in HTTP responses (e.g., 5xx errors).
    - Slow response times for critical services.
  - **Failover Events :**
    - Automatic notifications when a failover occurs in the floating IP or database cluster.


Float IP
keepalived
Swarm Cluster
SWARM API
Revers-proxy / Static File Server
sentry
Cultural Affairs WebAPP
database
redis
MaxScale Database Loadbalancer
Redis Cache
GlusterFs Shared Storage
deploy.bafar.iau.ir
mon.bafar.iau.ir
reg.bafar.iau.ir
sentry.bafar.iau.ir
Database Cluster
DB1
MariaDB Master
DB2
DB3
MariaDB Slave
MariaDB Slave
ManageServer
WebHook
Ansible Deployer
Monitoring stack
Registry stack
Sentry
Obsidian