BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Work Integrated Learning Program

M. Tech. Software Engineering

Software Architecture - SEZG651

Assignment 2 – Harbor

Name – Hemant Tiwari

BITS Student ID – 2022MT93184

Semester – 1st FY 2022-2023

Harbor - Introduction

- **Harbor** is an **open-source** trusted **cloud native registry** project that **stores, signs, and scans content**.

- Harbor extends the open-source Docker Distribution by adding the functionalities usually required by users such as security, identity and management. Having a registry closer to the build and run environment can improve the image transfer efficiency. Harbor supports replication of images between registries, and offers advanced security features such as user management, access control and activity auditing.

- Harbor is hosted by the **Cloud Native Computing Foundation (CNCF)**.

Features -

- **Cloud Native Registry:** With support for both container images and Helm charts, Harbor serves as registry for cloud native environments like container runtimes and orchestration platforms

.

- **Role based access control:** Users access different repositories through 'projects' and a user can have different permission for images or Helm charts under a project.

- **Policy based replication:** Images and charts can be replicated (synchronized) between multiple registry instances based on policies with using filters (repository, tag and label). Harbor automatically retries a replication if it encounters any errors. This can be used to assist load balancing, achieve high availability, and facilitate multi-datacenter deployments in hybrid and multi-cloud scenarios.

- **Vulnerability Scanning:** Harbor scans images regularly for vulnerabilities and has policy checks to prevent vulnerable images from being deployed.

- **LDAP/AD support:** Harbor integrates with existing enterprise LDAP/AD for user authentication and management and supports importing LDAP groups into Harbor that can then be given permissions to specific projects.

- **OIDC support:** Harbor leverages OpenID Connect (OIDC) to verify the identity of users authenticated by an external authorization server or identity provider. Single sign-on can be enabled to log into the Harbor portal.

- **Image deletion & garbage collection:** System admin can run garbage collection jobs so that images(dangling manifests and unreferenced blobs) can be deleted, and their space can be freed up periodically.

- **Notary:** Support signing container images using Docker Content Trust (leveraging Notary) for guaranteeing authenticity and provenance. In addition, policies that prevent unsigned images from being deployed can also be activated.

- **Graphical user portal:** User can easily browse, search repositories and manage projects.

- **Auditing:** All the operations to the repositories are tracked through logs.

- **RESTful API:** RESTful APIs are provided to facilitate administrative operations and are easy to use for integration with external systems. An embedded Swagger UI is available for exploring and testing the API.

- **Easy deployment:** Harbor can be deployed via Docker compose as well Helm Chart, and a Harbor Operator was added recently as well.

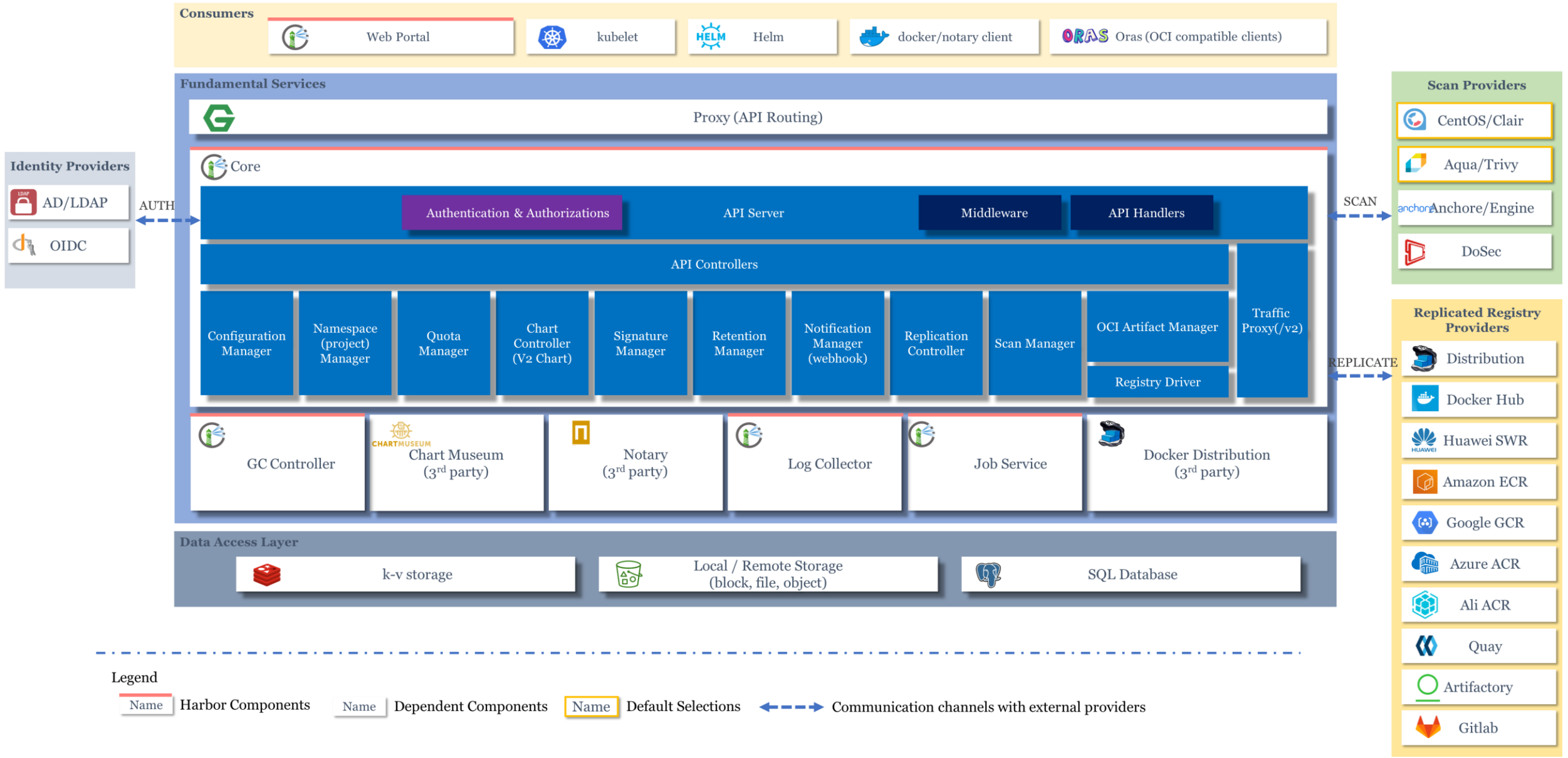
Utility Tree of the Architecturally Significant Requirements (ASRs)

Quality Attribute	Attribute Refinement	Scenario	Business Impact	Architecture Impact
Security	Accessibility	Managing the authentication, authorization of Artifacts.	High	High
	Middleware	Pre-process some requests in advance to determine whether they match the required criteria and can be passed to the backend components for further processing or not.	Medium	High
	API Handlers	Handle the corresponding REST API requests, mainly focus on parsing and validating request parameters, completing business logic on top of the relevant API controller, and writing back the generated response	High	High
	Auditing	All the operations to the repositories are tracked through logs.	High	High
Interoperability	Notification	Mechanism to send real time notification to users for any new changes in the artifacts	High	Medium
	Retention	Manages the tag retention policies and perform and monitor the tag retention processes	Medium	Medium
	Replication	Images and charts can be replicated (synchronized) between multiple registry instances based on policies with using filters (repository, tag and label)	Medium	High
Performance	Reliability	Scanning of the images for the vulnerabilities and has policy checks to prevent vulnerable images from being deployed	High	High
	Content Trust	Support signing container images using Docker Content Trust (leveraging Notary) for guaranteeing authenticity and provenance.	High	Medium
	Scalability	System should support big (larger size) data ingestion and should be scalable further	High	Medium
	Availability	System should support 99.999% Availability	High	Medium

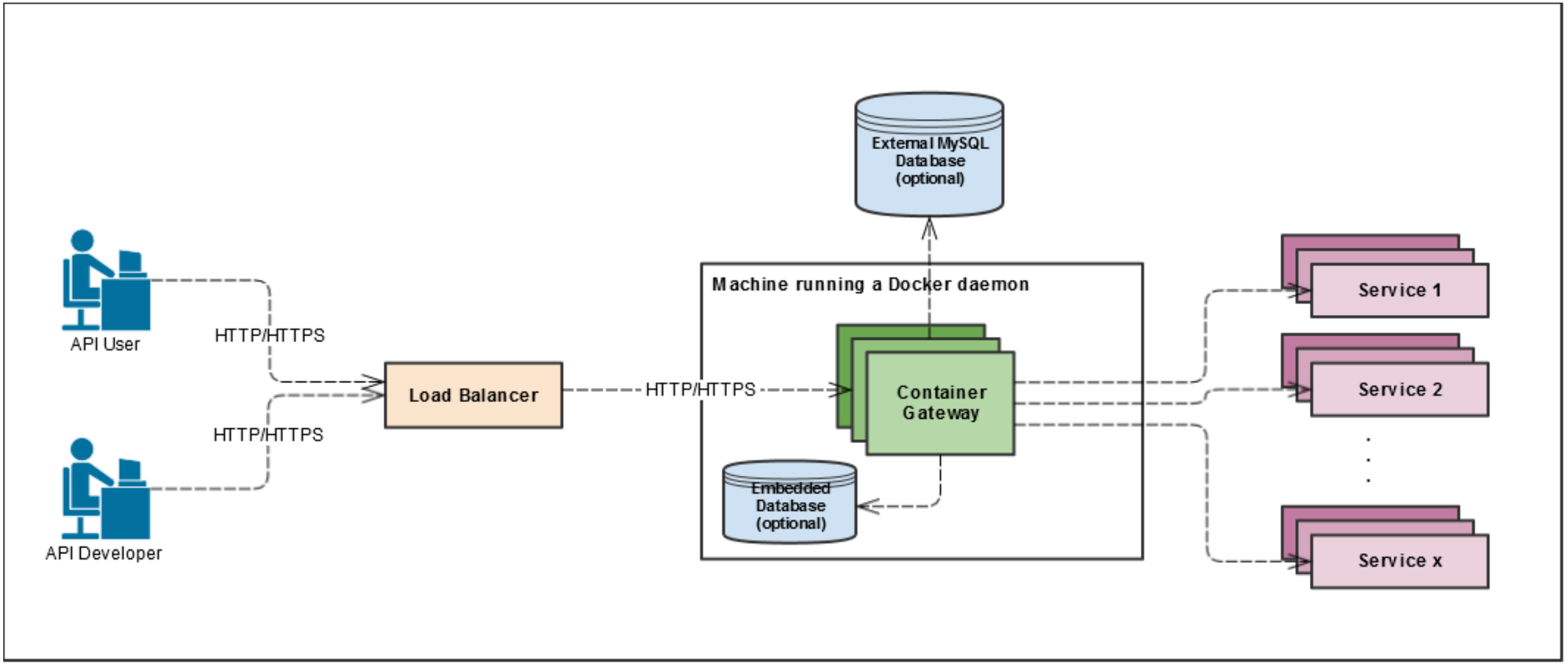
Tactics used to Achieve the ASRs

Quality Attribute	Tactic Used
Security	Requests are protected by the authentication service which can be powered by a local database, AD/LDAP or OIDC
	RBAC mechanism is enabled for performing authorizations to the related actions, e.g.: pull/push an image
	Token service is designed for issuing a token for every docker push/pull command according to a user's role of a project. If there is no token in a request sent from a Docker client, the Registry will redirect the request to the token service.
	Some functions are implemented as kinds of middleware, such as 'quota management', 'signature check', 'vulnerability severity check' and 'robot account parsing' etc.
	Handle the corresponding REST API requests, mainly focus on parsing and validating request parameters, completing business logic on top of the relevant API controller, and writing back the generated response.
	General job execution queue service to let other components/services submit requests of running asynchronous tasks concurrently with simple restful APIs
	Log collector, responsible for collecting logs of other modules into a single place.
	Harbor leverages OpenID Connect (OIDC) to verify the identity of users authenticated by an external authorization server or identity provider. Single sign-on can be enabled to log into the Harbor portal
Interoperability	Covers the management of all the system configurations, like authentication type settings, email settings, and certificates, etc.
	A mechanism configured in Harbor so that artifact status changes in Harbor can be populated to the Webhook endpoints configured in Harbor. The interested parties can trigger some follow-up actions by listening to the related webhook events. It is supported via HTTP Post request and Slack channel
	Manages the tag retention policies and perform and monitor the tag retention processes
	Manages the replication policies and registry adapters, triggers and monitors the concurrent replication processes. Many registry adapters are implemented such as, Distribution (docker registry), Docker Hub, Amazon ECR, Google GCR, Azure ACR, Helm Hub, Artifactory, GitLab Registry, etc.
Performance	A 3rd party content trust server, responsible for securely publishing and verifying content. In addition, policies that prevent unsigned images from being deployed can also be activated.
	System admin can run garbage collection jobs so that images(dangling manifests and unreferenced blobs) can be deleted, and their space can be freed up periodically.
	Manages the multiple configured scanners adapted by different providers and also provides scan summaries and reports for the specified artifacts. The Trivy scanner provided by Aqua Security, the Anchore Engine scanner provided by Anchore, the Clair scanner sponsored by CentOS (Redhat), and DoSec Scanner provided by DoSec will be supported.
	Proxy the chart related requests to backend chart museum and provides several extensions to improve chart management experiences.
	Manages the quota settings of projects and performs the quota validations when new pushes happened.
	Manages the base data and corresponding metadata of the project, which is created to isolate the managed artifacts.

Component & Connection View



Deployment View

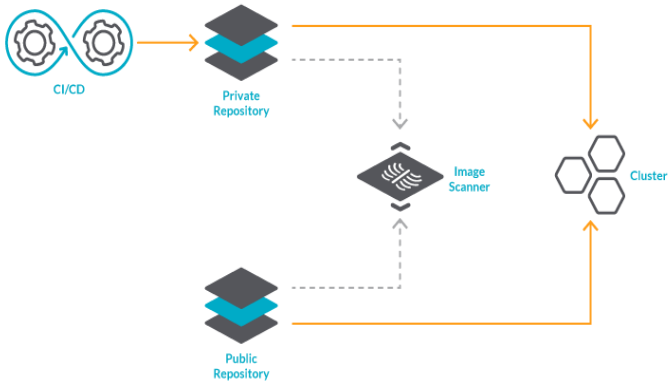


Sequence Diagram

Use Case - Scanning and pushing the image to Harbor Registry

Building the image using CI/CD and Pushing it to Harbor

Scanning Step



Architecture Patterns used in Harbor

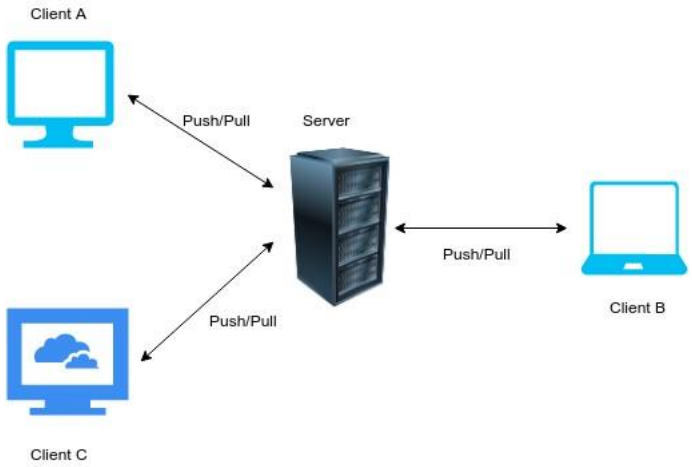
Harbor can be accessed via,

i. CLI (Command Line Interface)

ii. UI (User Interface)

For **CLI**, Harbor follows **Client-Server Pattern.**

We have multiple users (Clients) who push and/or pull images, charts, etc. from the Server using docker/helm commands.



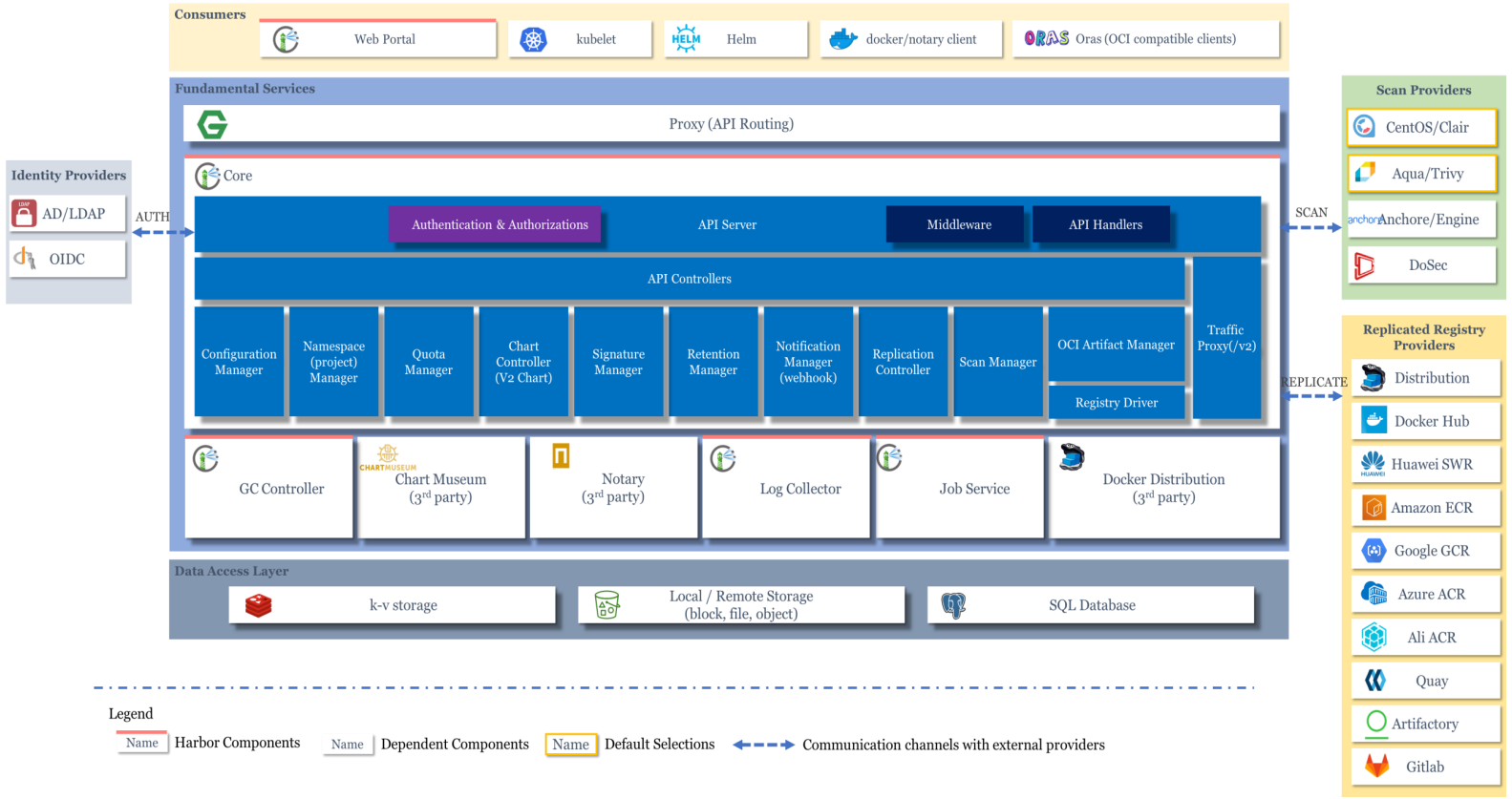
Client-Server Pattern

Architecture Patterns used in Harbor

For **UI**, Harbor follows **Layered Pattern.**

As it can be observed from the Component & Connection View below. We have,

1. **Presentation Layer** (also known as **UI Layer**)
2. **Application Layer** (also known as **Service Layer**)
3. **Business Logic Layer** (also known as **Domain Layer**)
4. **Data Access Layer** (also known as **Persistence Layer**)



Key Learnings from the Assignment

Following are my key learnings from the assignment,

- Importance of Architectural Diagrams, ASRs and Utility tree, and Architecture Patterns in Software Architecture.
- How architecture patterns help is categorizing the advantages and disadvantages of a system and help in improving the overall software architecture of an Application.
- ASRs and Utility trees help us in Determining the Impact of ASRs (User Stories) on Business and Architecture and prioritising the work for the application Development.

Thank You