documentation and dont skip anything.

Of course. Based on the extensive "Resource Configuration Overview" knowledge transfer session, here is a comprehensive and detailed documentation covering every mentioned aspect of the Kubernetes (Helm) configuration and deployment process.

**Kubernetes & Helm Deployment Configuration Guide**

**1. Overview & State of the System**

This document details the third and final phase of the deployment process: configuring and deploying the application to the Kubernetes (K8s) cluster. At this stage, the foundational Azure resources (VMs, Networks, Storage) have been **provisioned** by Terraform and **configured** (e.g., Oracle DB users, passwords) by Ansible.

* **Current State**: The infrastructure is ready and waiting. The Oracle database is running, with users and passwords configured. The application images (custom and out-of-the-box) are built and stored in a container registry.
* **Goal of this Phase**: To deploy the Windchill application onto the shared AKS cluster, telling it how to connect to the prepared resources (like the DB), how to scale, and how to handle traffic.

**Crucial Note:** The Terraform and Ansible repos are **only used for environment creation or recreation**. Routine application deployments, customizations, defect fixes, or updates **do not** involve these repos. This phase, using Helm, is for all ongoing deployment activities.

**2. Repository Structure: Helm Charts**

The configuration for deploying to Kubernetes is managed in a Helm chart repository (e.g., windshield-helm-charts).

**2.1. Key Directories and Files:**

* /charts/ambassador: Contains Ambassador (API Gateway / Ingress controller) configuration rules for different use cases:
  + no-sso.yaml: For login without Single Sign-On (e.g., using Kickstart).
  + vm.yaml: For local development environments running on a VM with a microk8s cluster.
  + cluster.yaml: The standard configuration used for the shared AKS cluster environments (dev, QA, prod).
* /charts/ingress: Configuration for WBM functionalities related to ingress.
* /charts/ldap: Configuration templates for OpenDJ (LDAP).
* /charts/windshield/templates/: **This is a critical directory.**
  + Contains standard Kubernetes YAML templates (e.g., deployment.yaml, statefulset.yaml, configmap.yaml) that define the structure of Pods, Services, etc.
  + **Changes here affect ALL environments.** Any modification requires a Pull Request (PR) and thorough validation, as it impacts the base definition for every deployment. These templates are typically updated only for major CPS upgrades or critical patches, often based on templates provided by PTC and adapted by the team.
* /releases/: Contains environment-specific value files.
  + **Structure**: releases/<team-name><environment-number>/ (e.g., releases/za01/, releases/psc01/).
  + Each folder contains a values.yaml file that customizes the base templates for its specific environment.

**3. Environment Configuration (**values.yaml**)**

The values.yaml file in each release directory is the primary point of configuration for an environment. It defines parameters that are injected into the Helm templates.

**3.1. Image Configuration**

* **Out-of-the-Box (OOTB) Images**: Provided by PTC and pulled from their Docker repository. These are listed in the images: section.
* **Custom Images**: Built from the three main repos (windshield-application, windshield-configuration, windshield-test-data). Their hashes (git commit SHAs) are specified to ensure a reproducible build.
  + scHash: Hash for the configuration repo.
  + mavenHash: Hash for the application/customization repo.
  + tdHash: Hash for the test data repo.
* **Image Types**:
  + image: The main Windchill application image.
  + nginxImage: The custom NGINX image for traffic routing.

**3.2. Application Parameters**

* infType: A crucial parameter (e.g., dev, prod, int). This value is passed to the application and determines which set of configuration files from the windshield-configuration repo are loaded during startup (e.g., which build.xml targets are executed). It defines the environment's behavior and loaded modules.
* version: The CPS version (e.g., CPS 5). This is automatically managed by workflows.
* utility:
  + enabled: Typically set to false and enabled by workflows during deployment.
  + resources: CPU and memory requests/limits for the "utility" pod. This pod is responsible for running BMW customizations and executing the build.xml process to prepare the database.

**3.3. Resource Definition & Scaling**

This section defines the size and scale of each application component pod.

* methodserver: Configuration for the main method server pods.
  + replicaCount: Number of pod replicas.
  + resources: CPU and memory requests and limits.
  + livenessProbe: Health check configuration.
* backgroundmethodserver: A **list** of configurations for background method servers. Different environments can have a different number of these servers (e.g., dev might have 1, prod might have 13). Each entry in the list defines:
  + name
  + queueGroup
  + resources
* servermanager: Configuration for the server manager pod (resources, livenessProbe).

**3.4. Secrets Integration**

* The values.yaml file does not contain actual secrets. It contains **references** to secrets.
* Example: dbPasswordSecret: db-password. This tells Helm to look for a Kubernetes Secret named db-password and use a specific key from it.
* **Consistency is Critical**: The secrets defined in the encrypted secrets.yaml file (e.g., db-password) **MUST** match the passwords configured in the Oracle database by Ansible. A mismatch will cause authentication failures and pod crashes.

**3.5. Database Connection**

* dbHost: The IP address of the Oracle VM (configured in Terraform/Ansible).
* dbService: The Oracle service name.
* dbPort: The database port.
* These parameters tell the application pods where to find the database.

**3.6. Optional Components**

* **SOLR Search (**solr**)**:
  + enabled: Set to true or false to deploy or not deploy the SOLR pod.
  + image: Uses an out-of-the-box PTC image.
  + resources: CPU/Memory for the SOLR pod.
  + **Prerequisite**: The persistent disk for SOLR must have been provisioned by Terraform.
* **LDAP (**ldap**)**:
  + serviceName: Points to the LDAP service running in the cluster (typically openDJ).
  + LDAP runs in a separate namespace (e.g., za-components) to ensure it remains running during application restarts, as the application requires LDAP to be available to start up.

**3.7. Ingress & Authentication**

* ingress:
  + host: Defines the public URL for the environment (e.g., za01-aks.windshield.example.com).
  + This works in tandem with the Ambassador cluster.yaml rules.
* tokenConfiguration: Contains settings for Webnex authentication, defining the OAuth flow and redirect URLs for SSO.

**3.8. Bulk Migrator (WBM)**

* wbm:
  + enabled: Typically set to true by workflows when a WBM job needs to run.
  + image: Specifies the custom WBM image.
  + resources: CPU/Memory allocated for WBM jobs.
  + Can use a dedicated database user (configured in Ansible).

**4. Secrets Management with SOPS**

Secrets are stored in an encrypted secrets.yaml file within each environment's release directory (e.g., releases/za01/secrets.yaml).

* **Purpose**: Stores all sensitive data (passwords, tokens, etc.) encrypted at rest.
* **Technology**: Uses **SOPS** (Secrets OPerationS) for encryption/decryption.
* **Contents**:
  + Database passwords (dbaPassword, dbPassword, wbmPassword)
  + LDAP bind passwords
  + Windchill user passwords
  + API tokens (e.g., for Webnex)
* **Process**:
  1. A master SOPS key is managed securely (e.g., in Azure Key Vault).
  2. To edit secrets, a user must decrypt the file locally using the key: sops --decrypt secrets.yaml > decrypted\_secrets.yaml
  3. After editing, encrypt the file again: sops --encrypt decrypted\_secrets.yaml > secrets.yaml
* \*\* synchronization\*\*: The values in this file **must be synchronized** with the values configured by Ansible in the Oracle database.

**5. Workflow: How It All Fits Together**

1. **Deployment Command**: A command like helm upgrade -f releases/za01/values.yaml -f releases/za01/secrets.yaml windshield ./charts/windshield is executed (manually or via CI/CD).
2. **Template Rendering**: Helm takes the generic templates from /templates/ and injects the environment-specific values from values.yaml and secrets.yaml.
3. **Kubernetes Manifest Creation**: Helm generates final Kubernetes YAML manifests.
4. **Application to Cluster**: Helm communicates with the AKS cluster and applies these manifests, resulting in:
   * Pods being created (methodserver, backgroundmethodserver, servermanager, optionally solr).
   * Services being exposed.
   * Ingress rules being configured via Ambassador.
5. **Pod Startup**: When a pod starts, it:
   * Reads its configuration from mounted configMaps (built from values.yaml).
   * Pulls secrets from the Kubernetes secrets API (decrypted from secrets.yaml).
   * Connects to the Oracle DB using the hostname and password from these configurations.
   * Loads its configuration based on the infType parameter.

**6. Namespace Strategy**

* <team>-windshield (e.g., za-windshield): Contains the main application pods (methodserver, backgroundmethodserver, servermanager). These can be restarted frequently.
* <team>-components (e.g., za-components): Contains long-lived, foundational services like **LDAP (**openDJ**)**. This namespace is kept separate to ensure these critical services are not accidentally restarted or deleted when the application is redeployed, ensuring higher availability and stability.

This structured approach ensures a clear separation of concerns, manageability, and resilience for the deployed application environments.