Kong Hybrid Deployment - Migrating to PostGres Database

The migration to a database-backed Kong API Gateway is a critical step for enabling hybrid deployment, providing a flexible architecture that offers significant benefits, such as reduced direct traffic to the database by separating control and data planes, allowing for efficient data plane management across diverse environments. This separation not only enhances performance by distributing traffic more effectively but also increases security by isolating the control plane layer from the data plane layer.

So first step is the migration process of our Kong API Gateway from a DB-less configuration to a database-backed (PostgreSQL) deployment within our Kubernetes cluster. We choose DM as the environment for the POC. This migration enables the utilization of Kong Manager, Kong Admin API, Kong Portal, and Single Sign-On (SSO) through Okta. The process involves configuring the necessary components, migrating configurations, and ensuring security and ease of management through centralized authentication and authorization.

Goals

- Migrate Kong API Gateway to a database-backed configuration.
- Integrate Kong Manager, Admin API, and Kong Portal for enhanced API management and user experience.
- Implement SSO using Okta for secure and centralized user authentication.

Initial Configuration: DB-less Deployment

Our initial setup was based on a DB-less configuration for Kong API Gateway, streamlined for Kubernetes deployment. The configuration focused on a lightweight, declarative approach to manager ingress and service configurations.

The migration to a database-backed deployment involved several critical steps, including configuring PostgreSQL as the backend database and adjusting Kong's deployment to interact with this database. Postgres sub chart is added for this purpose, which will configure a database with PVC and PV configuration.

Database Configuration

- **PostgreSQL**: Enabled as the backend database for Kong, with credentials and connection details securely managed.
-
```
env:
  database: "postgres"
  # kong_admin is the user name here, (if we are using basic authentication)
  password:
    valueFrom:
      secretKeyRef:
          name: kong-enterprise-superuser-password
          key: password
postgresql:
  enabled: true
  auth:
    username: "kong"
    database: "kong"
    existingSecret: "kong-datamigration-postgresql"
    secretKeys:
      adminPasswordKey: "postgres-password"
      userPasswordKey: "password"
```

- **Kong Configuration**: Updated to interact with PostgreSQL, including adjustments to environment variables and deployment parameters.

Enhanced Features Integration

Kong Manager

- **Enabled**: Ingress and protocols are created via values.yaml file itself. Separate ingress definition is not needed.

```
 rbac:
   # Specifies whether RBAC resources should be created
   create: true
```

```
manager:
  enabled: true
  annotations:
    konghq.com/protocol: https
  type: ClusterIP

  tls:
    enabled: true
    servicePort: 8445
    containerPort: 8445
    parameters:
    - http2

  ingress:
    enabled: true
    ingressClassName: kong-datamigration
    hostname: kongmanager.dev2.pickles.com.au
    path: /

```

- **Authentication**: Integrated with Okta for SSO, using OpenID Connect for secure access.enterprise:

```
enabled: true
  license_secret: kong-enterprise-license
  vitals:
    enabled: true
  portal:
    enabled: true
  rbac:
    enabled: true
    admin_gui_auth: openid-connect
    session_conf_secret: kong-session-config
    admin_gui_auth_conf_secret: kong-oidc-configuration
```

- **Authentication**: If we are planning to use basic authentication (username and password without SSO), use below:

```
enabled: true
  license_secret: kong-enterprise-license
  vitals:
    enabled: true
  portal:
    enabled: true
  rbac:
    enabled: true
    admin_gui_auth: basic-auth
    session_conf_secret: kong-session-config
```

Kong Admin API

- **Accessibility**: Exposed securely through ingress configurations via values.yaml, with HTTPS enforced. Like Kong Manager, both Manager and Admin API are exposed via same domain.

```
admin:
  enabled: true
  type: ClusterIP
  annotations:
    konghq.com/protocol: https

   tls:
    enabled: true
    servicePort: 8444
    containerPort: 8444
    protocol: TCP
    parameters:
    - http2

  ingress:
    annotations:
      konghq.com/strip-path: "true"
    enabled: true
    ingressClassName: kong-datamigration
    hostname: kongmanager.dev2.pickles.com.au
    path: /api
```

Kong Ingress Controller Configuration: Superuser Password and Admin Token should be same

When deploying Kong using the official Helm chart, there are two important configuration values related to authentication:

1. `env.password`: This value sets the password for the Kong Enterprise superuser account (`kong_admin`). The superuser account is a special administrative account that has full access to the Kong Enterprise system and is used for initial setup, configuration, and management.
2. `ingressController.env.kong_admin_token`: This value represents the authentication token used by the Kong Ingress Controller to communicate with the Kong Admin API. The Kong Ingress Controller is responsible for managing and configuring Kong based on Kubernetes Ingress resources.

By default, the Kong Helm chart sets these two values to be the same for convenience. When Kong is first installed, the `kong_admin` bootstrap user is the only admin user available, and its basic-auth password and RBAC token are set to the same value specified by `env.password`. This allows the Ingress Controller to work out of the box without additional configuration.

However, it's important to note that the `ingressController.env.kong_admin_token` value can be set to any valid RBAC token, as long as the associated roles have sufficient permissions to program the required entities in the Kong Admin API.

If the values of `env.password` and `ingressController.env.kong_admin_token` are not set to be the same, the Kong Ingress Controller pod will fail to start with the following error message:

- ```
  time="2024-04-01T13:58:27Z" level=info msg="diagnostics server disabled"

  time="2024-04-01T13:58:27Z" level=info msg="starting controller manager" commit=956f457aafc2a07910bfa3c496e54

  time="2024-04-01T13:58:27Z" level=info msg="the ingress class name has been set" logger=setup value=kong-data

  time="2024-04-01T13:58:27Z" level=info msg="starting standalone health check server" logger=setup

  time="2024-04-01T13:58:27Z" level=info msg="getting the kong admin api client configuration" logger=setup
  ```
  ```

```
10
11 time="2024-04-01T13:58:27Z" level=info msg="Retrying kong admin api client call after error" error="HTTP stat
```

This error occurs because the Kong Ingress Controller uses the `ingressController.env.kong_admin_token` value to authenticate with the Kong Admin API. If this token doesn't match the password set for the Kong Enterprise superuser account (`env.password`), the Ingress Controller will fail to authenticate and receive a 401 Unauthorized error.

Here's an example of how these values can be configured in the Kong Helm chart:

- Secure Admin API using token. and setting up super user password

```
 env:
  password:
    valueFrom:
      secretKeyRef:
        name: kong-enterprise-superuser-password
        key: password

ingressController:
  env:
    kong_admin_token:
      valueFrom:
        secretKeyRef:
          name: kong-admin-token-secret
          key: token
```

To ensure smooth operation of the Kong Ingress Controller and avoid authentication errors, it is essential to set the values of `env.password` (Kong Enterprise superuser password) and `ingressController.env.kong_admin_token` (Kong Ingress Controller authentication token) to be the same when deploying Kong using the Helm chart. Also, It's crucial to ensure that these values are kept secure and not exposed publicly. If the Ingress Controller token is compromised, it could allow unauthorized access to the Kong Admin API.

Kong Portal Setup

- **Enabled**: True, providing a developer portal for API documentation and management.
- **Authentication**: Configured for secure access, integrated with the broader SSO strategy.

SSO via Okta

- **Configuration**: Utilized Okta for SSO across Kong Manager and Kong Portal, ensuring secure and centralized user authentication. We can use Kong OpenId Connect plugin like below to enable authentication for Kong Manager:

```
apiVersion: configuration.konghq.com/v1
kind: KongPlugin
metadata:
  name: okta-kongmanager
  namespace: kong-datamigration
config:
  auth_methods:
    - authorization_code
    - session
  client_id:
    - REDACTED
  client_secret:
    - REDACTED
  issuer: https://pickles.oktapreview.com/oauth2/default
  redirect_uri:
    - https://kongmanager.dev2.pickles.com.au
  session_secret: REDACTED
plugin: openid-connect
```

- But by using the configuration below in values.yaml file referring the secret kong-oidc-configuration (details will be same as the above plugin but configured as a secret), OpenID Connect authentication will be enabled for Kong Manager. rbac:

```
    enabled: true
    admin_gui_auth: openid-connect
    session_conf_secret: kong-session-config
    admin_gui_auth_conf_secret: kong-oidc-configuration
```

- It is unnecessary to manually enable the OpenID Connect plugin via Admin API or Kong Manager.Below is the secret and value of the encoded string. (Enable OIDC for Kong Manager - Kong Gateway | Kong Docs)

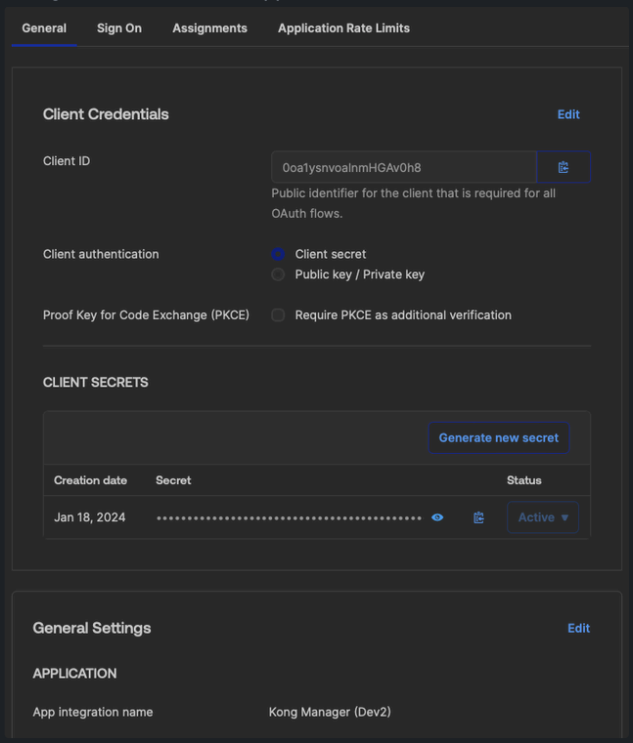
```
apiVersion: v1
kind: Secret
metadata:
  name: kong-oidc-configuration
  namespace: kong-datamigration
data:
  admin_gui_auth_conf: >-
    base 64 encoded string
type: Opaque

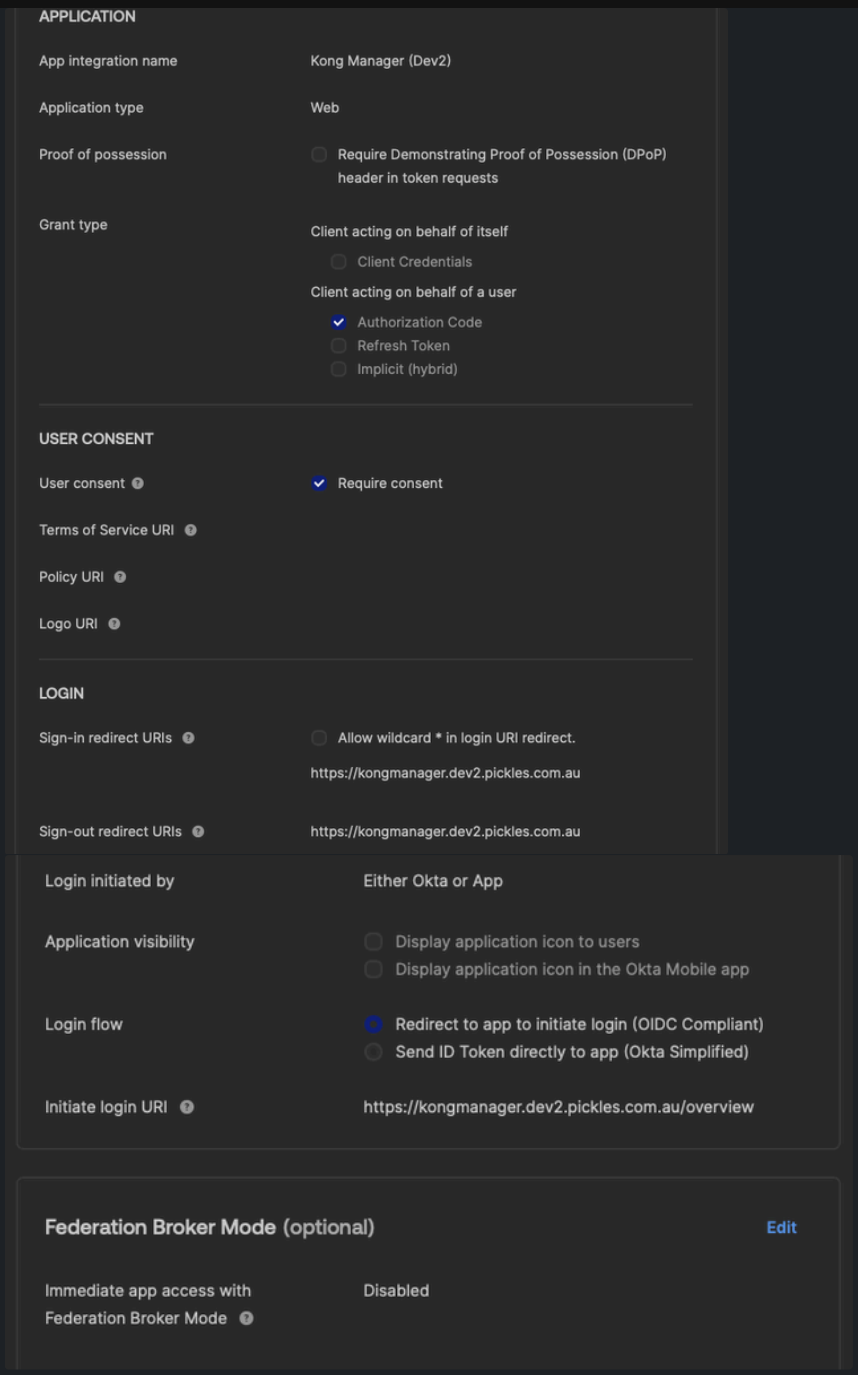
# Value of the encoded string
{
    "issuer": "https://pickles.oktapreview.com/oauth2/default",
    "admin_claim": "email",
    "client_id": [
        Redacted
    ],
    "client_secret": [
        Redacted
    ],
    "authenticated_groups_claim": [
        "groups"
    ],
    "ssl_verify": false,
    "leeway": 60,
    "redirect_uri": [
        "https://kongmanager.dev2.pickles.com.au"
    ],
    "login_redirect_uri": [
        "https://kongmanager.dev2.pickles.com.au"
    ],
    "logout_methods": [
        "GET",
        "DELETE"
    ],
    "logout_query_arg": "logout",
    "logout_redirect_uri": [
        "https://kongmanager.dev2.pickles.com.au"
    ],
    "scopes": [
        "openid",
        "email"
    ],
    "admin_auto_create_rbac_token_disabled": false,
    "admin_auto_create": true
}
```

- **Roles and Permissions**: Custom API call implemented to designate specific users as super-admins, leveraging the Kong Admin API for role assignment.

Security and Authentication

- **Secrets Management**: Utilised Kubernetes secrets for secure storage of sensitive configurations, including Okta client credentials and Kong's superuser password.
- **Okta Integration for RBAC**: The secret `kong-oidc-configuration` contains the OIDC configuration for integrating Kong with Okta for SSO.
- **Session Configuration for Admin GUI**: The `kong-session-config` secret holds the session configuration for the Kong Admin GUI.
- **Superuser Password**: The `kong-enterprise-superuser-password` secret contains the password for the superuser account, critical for initial setup and secure access.
- Okta configuration can be manipulated using following link. https://pickles-admin.oktapreview.com/. and below are the major configuration for the Okta application.



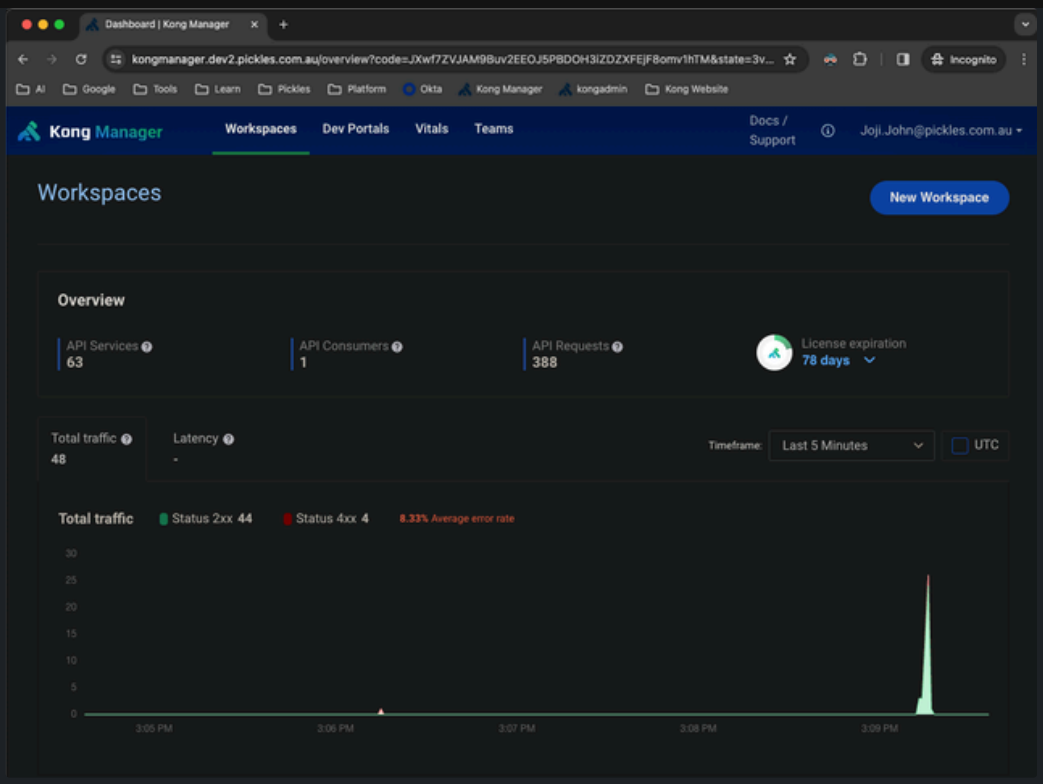


Conclusion

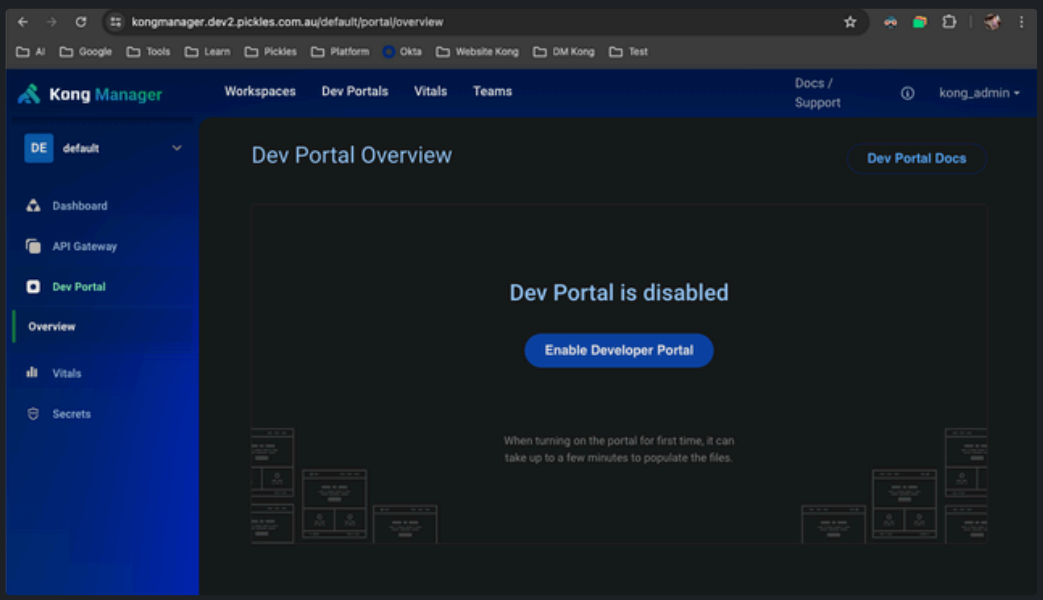
This needs a clean installation, if in between database is already created, we need to remove the PVC (as it is a POC, or else this needs careful migration). After completing the installation, you have the capability to promote an Okta user to a super admin role. This elevated permission allows the designated user to subsequently create and manage additional users within the system. To assign super admin privileges to an Okta user, utilise the following cURL command:

```
curl --location 'https://kongmanager.dev2.pickles.com.au/api/admins/Joji.John@pickles.com.au/roles' \
--header 'Kong-Admin-Token: {{Use the token from K8s secret kong-admin-token-secret}}' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'roles=super-admin'
```

Following this, when accessing Kong Manager using the Okta credentials, it will be redirected to the Manager dashboard with super admin permission. User can access the teams tab to create additional users and access.



Even though the portal is enabled in chart, we needed to enable kong portal again through Kong Manager and it is a one time activity.



This migration and enhancement of Kong API Gateway not only improved our API management infrastructure's scalability and security but also enriched the feature set with Kong Manager, Admin API, Kong Portal, and integrated SSO via Okta. The highlighted configuration snippets and the use of Kubernetes secrets for secure configuration management were pivotal in this process.