developer-experience-2.zip

CONTENTS

1. Developer Experience

1.1. APIs

Kyndryl Developer Experience (KDE) has a centralized location to discover, manage and run all APIs required to perform all the activities necessary for a developer’s job. This will allow developers to collaborate on API development, share feedback, and contribute to API documentation. In this centralized location for KDE APIs, developers can easily onboard, view, and invoke new APIs. They also have complete visibility of the API repository, API owner, any tech docs associated with the repo/API, OpenAPI specifications and access the API documents available on Swagger. Alternatively, API owners can manage API versions and track changes from one single place.

KDE service APIs are displayed based on the role levels established by the KDE role and the teams that have been assigned to.

APIs can be searched using filtering and advanced search options based on various criteria, such as versions, teams, applications, components, created date, modified date, or tags. If the relevant API is not available, the following message will be displayed: "Sorry, there are no docs matching the selected criteria"

A swagger button is available for each row that will redirect you to the swagger page in a new tab and display the APIs relevant to the developer's team or associated teams.

In KDE APIs swagger page, you can find the following information:

- API name (Title)
- API description
- API type
- API owner
- Tags
- Associated component
- Lifecycle: Development/ Alpha/ Beta/ In Production/ Deprecated/ Retired

From the KDE APIs page, you can perform the following actions:

- Invoke / run: Run the API as expected.
- View: View the list of APIs available for each user. This list will change depending on the user’s rights and areas of expertise.
- Edit: Based on the permissions granted, users can edit APIs accordingly. However, if users have edit permissions in GitHub repository, they can still update the API from the code on GitHub.

If you click the API menu on the left, it should redirect you to a page for Kyndryl's REST APIs. To add a new API, go to the **Component Inventory** page and use the **Onboard New Component** option.

API synchronization

KDE now enables automatic synchronization of API definitions from Swagger URLs instead of only relying on static YAML files. This ensures that KDE’s API Explorer and the VS Code custom extension reflect the most up-to-date API specifications when accessing Swagger without authentication.

For squads that maintain a Swagger file but require authorization, users can still view Swagger but will not be able to execute it.

Benefits:

Real-time accuracy

- When connected to Swagger hosted without authentication, the develop console retrieves the latest API specifications, ensuring developers work with the most up-to-date endpoints, parameters, and response schemas.
- Reduces the risk of outdated API documentation, a common issue with static YAML files.

Automated synchronization

- Eliminates the need for manual updates and version control of static files.
- Ensures all team members have access to the latest API specifications without additional effort.

Environment-specific configurations

- Dynamically adjusts API documentation based on environment settings (for example staging versus production base URLs).
- Reflects authentication-based access restrictions without requiring multiple YAML file versions.

API visualization in KDE

The **API visualization graph** provides a clear and structured way for developers to understand API dependencies and relationships within KDE. By introducing a dedicated API graph, users can better visualize how APIs are consumed and exposed, improving discoverability and usability.

The **API visualization graph** is integrated into the API Explorer (detailed view) and into the Component API tab. The default display is **left to right** for clarity.

API relationship and labels

Label	Description
Owner Of	Points from the owner to the components they manage.
Owned By	Points from a component back to the accountable owner.
ProvidesApi	Links a component that exposes an API.
ProvidedBy	Points from an API back to the component exposing it.
ConsumesApi	Points from a component to the API it depends on.
apiConsumedBy	Points from an API back to the components using it.
dependsOn	Shows what an entity needs.
dependencyOf	Shows what depends on an entity.
parentOf / childOf	Represents hierarchical relationships between components.
memberOf / hasMember	Defines group memberships.
PartOf / hasPart	Represents sub-component relationships.

Icon representation

Name	Icon
Teams/Groups: Represents different development teams.	
System (Environment): Indicates environment-related components.	
API: Highlights APIs within the system.	
Component: Represents backend and microservices.	
Resources: Displays additional components relevant to APIs.	
Domain: Represents logical groupings of related services and APIs.	

1.2. How to create a new GEMU repository via KDE template

Kyndryl Developer Experience (KDE) provides a governed and automated workflow for creating GitHub Enterprise Managed Users (GEMU) repositories. This process ensures compliance with **Secure SDLC (S-SDLC)** standards, enforces organizational naming conventions, and integrates automated checks and approval mechanisms. Follow the steps mentioned below to create a new GEMU repository using the KDE template.

wrwrgwrgwrgwrg

Step 1: Access the KDE template

1. Navigate to **KDE Portal** >**Templates**
2. Select **Repo Creation Template**.
    - Available templates include
        - To create an empty default repository
        - To create a repository with boiler plate code for a specific technology
    - Each template comes with:
        - Boilerplate code
        - Pre-configured CI/CD pipelines
        - Embedded S-SDLC checks (branch protection, PR review enforcement).

Step 2: Enter required information

Provide the following details to create your repository. Some fields are auto-populated based on your profile.

- Virtual Organization (Mandatory)
- Team Name (Optional)
- Repository Name (Mandatory)
    - Example:
        - bdg-sw-devx-kde-test (CTO organization)
        - ac-project1-testrepo (Delivery organization)
- Justification
- State reason

Virtual Org naming convention

- Enforced format: **org-team-project**
- Supported Virtual Orgs:
    - CTO
    - Delivery
- Purpose:
    - Ensures traceability

- Prevents naming conflicts
- Supports compliance and audit requirements

Step 3: Approval workflow

When you create a repository, an automated approval process starts.

- **Level 1:** Your manager reviews and approves the request.
- You can view approval status and audit logs in KDE for traceability.

Step 4: Automated checks and governance

The template provides automated checks to help ensure security, compliance, and consistency across all repositories. These checks include:

- Branch protection rules
- Mandatory PR reviews
- Restrictions on force pushes
- Integrated Security Scans:
  - Mend.io for dependency checks
  - Image scanning for container security
- Pre-commit validation for compliance.

Step 5: Post-creation actions

After the repository is created:

- The repository is **auto-registered in KDE** as a software entity for visibility and governance.
- Audit logs are maintained for all actions.
- Access requests within CTO or cross-org are managed via KDE through a multi-stage approval process.

Key benefits

Using the KDE template for GEMU repository creation offers the following advantages:

- **Standardization:** Ensure consistent naming and structure across all GEMU repos.
- **Security:** Provides built-in S-SDLC controls and automated compliance checks.
- **Efficiency:** Uses pre-approved templates to reduce setup time and minimize errors.
- **Traceability:** Maintains a complete audit trail and governance visibility in KDE.

FAQs

1. Q: Are other Github operations changing due to repo creation via KDE? Ans: No, just the repository creation and repo access go through KDE. Other things are not changing and your Git access will not be revoked.
2. Q: How to bring modifications to the templates ? Ans: The KDE framework is extensible through custom templates that can be shared.
3. Q: Is there an API for repository creation ? Ans: Currently not avaliable. KDE API will be available shortly.

4. Q: Can Virtual Org owners and approvers be changed. Ans: Yes, **process is outlined** by the SSDLC team
5. Q: What happens if a GitHub rate limit is reached during repository creation—will the process fail gracefully, or could it result in a partially configured repository ? Ans: Any error, the flow fails cleanly without any partial set up.
6. Q: Is there any dashboard or place to see all repos created through KDE? Ans: Repos created via KDE are automatically registered as entities which can be viewed in Software Entities page
7. Q: If someone creates a repo by mistake, is there an easy way to delete or roll it back in KDE ? Ans: KDE does not support repo deletion. Needs to be done directly in Git

1.3. Docs

The Kyndryl Developer Console is a treasure trove of knowledge, allowing developers to search for a wide variety of tech documents (best practices, guidelines, reference architecture, troubleshooting guides, and Know-How, among others) to resolve inquiries during the development process. This diverse range of resources reduces time spent looking for documentation or creating duplicate documentation, making you feel more informed and resourceful.

The Docs page from the Kyndryl Developer Console provides centralized access to service onboarding guides, tech documentation, whitepapers, and best practices. Access the Developer Console and select the **Docs** tab to see the technical documentation.

This page contains all Markdown files from the Developer Console. It displays any Markdown pages associated with your teams based on **View** or **Edit** roles.

A table listing all the documentation available is presents the following details:

- Doc Name (Title)
- Doc Owner
- Doc Created Date
- Doc Last Modified Date
- Team Name
- Tags
- Associated Component
- Doc Source Link

To search for a specific tech doc, use the filter located at the left of the page. Here, you can also access your favorite documents ensuring a much quick access. Additionally, you can copy the link to a clipboard. You can also have access to your owned documentation from the filter, which allows you to quickly find and manage the documentation you are responsible for.

If no results are found, a message displays: "*Sorry, there are no docs matching the selected criteria*." If your search results take more than five seconds to load, a message displays: *“Documentation is accessed for the first time and is being prepared. The subsequent loads are much faster”*.

1.4. Home Page

The Developer Console home page allows developers to view pending tasks, commonly used developer tools, and access resources relevant to complete their work. A centralized page accelerates the developer's velocity and agility by allowing them to access Kyndryl SharePoint documents related to security open-source guidelines (SSDLC or OSSC guidelines) and development tools (ADO, GitHub, and Viva channel for CTO Developers) removing any distraction.

Only users with a Developer Viewer permission can access the Developer Console. For additional details, see the Roles page. To access the Developer Console, login to your Kyndryl Bridge CTO account. Then, go to the Bridge App menu and select the Developer Console option. The Home page displays within the same Kyndryl Bridge tab.

The following features can be found on this page:

1. **Azure DevOps (ADO) boards and widgets**

The Developer Console allows developers to quickly review and manage their tickets helping prioritize their work. From the Home page, you can view three subcomponents:

- **Plan My Day** widgets: At the top of the page, you will see six widgets. Here, only the data from boards that have been onboarded to the Kyndryl Developer Experience will be retrieved. If more than one board is subscribed to Kyndryl Developer Experience, the sum of all tickets from all boards will reflect on the widget. Click the **View Details** link to review the tickets displayed on the widgets. A new tab with the ADO details is displayed.

The information displayed in these components set of widgets is in real-time; you can review the current time at the top of the widgets. The total count of tickets includes all work item types and will exclude all the tickets in closed and removed status.

    - **Severity 1:** This widget retrieves the total count of all severity 1 active tickets currently available in your queue.
    - **Priority 1:** This widget retrieves the total count of all active tickets with a priority 1 available in your queue.
    - **Due Today:** This widget retrieves the total count of active tickets with today's due date currently available in your queue.
    - **Assigned Today:** This widget retrieves the total count of all active tickets that have been assigned to you today.
    - **In 'Active' Status:** This widget retrieves the total count of all tickets currently in active status that are available in your queue.
    - **In 'Ready' Status:** This widget retrieves the total count of tickets currently in ready status that are available in your queue.
- **My Tickets** table: In this table, you can find all your tickets from the boards onboarded to Kyndryl Developer Experience. By default, 5 tickets are displayed on the screen. You can change it to display 10 or 20. Click the **View all tasks** button to view all tasks under your name. If you need to look for a specific ticket, use the search bar to look for ID or title.
- **Squat Tickets** table: In this table, you can find all squat-related tickets from the boards onboarded to Kyndryl Developer Experience. By default, 5 tickets are displayed on the screen. You can change it to display 10 or 20. Click the **Ticket ID** to open the ticket in the ADO board. If you need to look for a specific ticket, use the search bar to look for ID or title. Use the dropdown menu to filter tickets by squat name. Click the **Squad ADO Board** link to open the bord in ADO.

1. **Developer Tools**

By having the developer tools in one single place with their respective icons and other quick links in one single section, the Developer Console home page reduces time spent on issue resolution and information gathering improving the productivity and reducing cognitive load; it also promotes a standardization of tools and technologies within the team.

Here are the tools that you can directly access from the Home page:

- **GitHub:** Click the link to open your GitHub page. This allows a direct access to your repositories and teams in GitHub.

- **JFrog Artifactory:** Click the link to open your JFrog Artifactory page. This allows a direct access to the antifactory.
- **DevOps Intelligence:** Click the link to be redirected to DevOps Intelligence overview where you can review the metrics at different stages of the CI/CD pipelines.
- **Community Engineering Portal:** The community engineering portal allows to search, reuse and contribute to code reusable templates within Kyndryl.
- **CTO Tech Forum:** This Viva Engage forum allows developers to ask questions and share relevant information enhancing collaboration, providing quick solutions to your questions, and facilitating the sharing of valuable information within the entire CTO technical community.
- **Visual Studio Code (VS Code):** An extension of VS Code is available allowing developers to navigate back from Visual Studio desktop version to the Software entities page on Developer Console where you can review components, associated tech docs, APIs and other details required for development. For Developer Console and GTM applications, you can be able to view the pipelines and view the APIs directly through the Swagger API so you can easily execute them.

1. **Quick links**

Here, you will find the most useful links for all developers in the community. Click the link to be redirected to the specific page or forum you would like to access. Click the arrow from the carrousel to review all the links available for you.

1. **Recently visited**

This is a list of all pages that you recently visited. To revisit any of these pages, click the correspondent link.

1. **Starred entities**

Here, you will store your most used entities. To add a new item, simply click the star icon next to the entity name in the Software Entities page.

1.5. Internal Use Only

YAML generator

YAML files are crucial in defining and managing software components, services, and catalog items. However, the manual creation of each file, especially given the number of YAML files required to configure the tools and plugins, can be time-consuming and challenging. Automating the creation of YAML files in KDE can significantly reduce developers' time on these repetitive tasks, freeing them to focus on more engaging activities like coding and feature development. This shift in focus leads to a more efficient use of time and resources, making the development process more streamlined and productive.

Administrators can use the YAML generator to generate files from existing data in source code repositories. These YAML files can typically have component metadata, including Service name, Owner/team information, Repository URL, Description, Tags, or labels for categorization. Scaffolding templates can automatically generate YAML files based on predefined configurations. YAML files can also source from API definitions, especially from OpenAPI. In some cases, it should be possible to generate YAML files based on the inputs developers manually provide through the User Interface (UI) or Command Line Interfaces (CLIs). A track includes fields like ownership, lifecycle status and versioning, which are converted into YAML format.

To create a YAML file, access the YAML generator tab from KDE. Select your entity type and add the metadata (name, title and description). Depending on the entity type (System, API or component), add the correspondent information:

- **System:** Include the owner.
- **API:** Include the API type (OpenAPI, AsyncAPI, gRPC, or GraphQL), API definition, lifecycle and system.

Component: Include the owner, component type, lifecycle, parent components, provided and consumed APIs, and dependencies.

Once all details are completed, generate the YAML descriptor for it. Copy the YAML text to the clipboard and paste it into a new file in an editor. Save the file as catalog-info.yaml, and place the file at a location from where KDE can discover it.

Roles for Kyndryl Developer Experience

Only developers who are onboarded and mapped to a specific Bridge Account and assigned the **Bridge Viewer** role, are be able to view the Developer Console option under the Bridge Main Menu. Clicking the Developer Console option from Bridge opens up the Console in the same screen.

The access group set in Bridge IAM only allows you to view the Developer Experience from the Console menu. However, the KDE roles established using the GitHub organization permissions will allow users to perform the different activities required for their job.

The following roles are supported for Kyndryl Developer Experience:

Users related to	KDE role (set internally)	Description
KDE Platform team	Super Admin role	All KDE permissions granted.
Dev Manager or Architect	Team Admin role	Can perform the following actions within the team or to (selective) teams: • Add components • Delete components • Add services • Add APIs
Developers	Operator role	• Can use software templates • Can add or delete their own components • Can add or delete their own services
Leadership team	Auditor role	Can get a comprehensive view of: • The entire CTO organization • Services components of different teams
Viewers	Viewer role	• Can only view access components of own organization • Can only view exposed components of other organizations. **Note:** The GitHub repository permissions may precede the viewer role allowing users to edit when accessing the service/component from GitHub.

Adding KDE roles:

1. Log in to Kyndryl Bridge
2. Access the IAM page
3. Click **Add New**.
4. Click Add next to the permission of your choice and see the Summary pane for confirmation.
5. Once you are satisfied with your selection, click **Add** at the bottom of the page to finish. The new custom role is created and displayed under the **Roles** tab.

1.6. KDE Self-Service Onboarding

Overview

The KDE (Kyndryl Developer Experience) Self-Service Onboarding feature enables squad leaders to independently onboard and manage their squads, systems, components, APIs, and related resources. This unified and guided interface minimizes dependency on platform administrators, standardizes onboarding workflows, and significantly accelerates time-to-value for engineering teams. This document covers the following sections:

1. **Prerequisites**
2. **Squad Onboarding**
3. **Squad Management**
4. **System & Entity Onboarding**

Prerequisites

Before starting the self-service onboarding process, ensure the following prerequisites are met:

- You have access to the KDE Portal.
- You have a valid enterprise user account (corporate email).

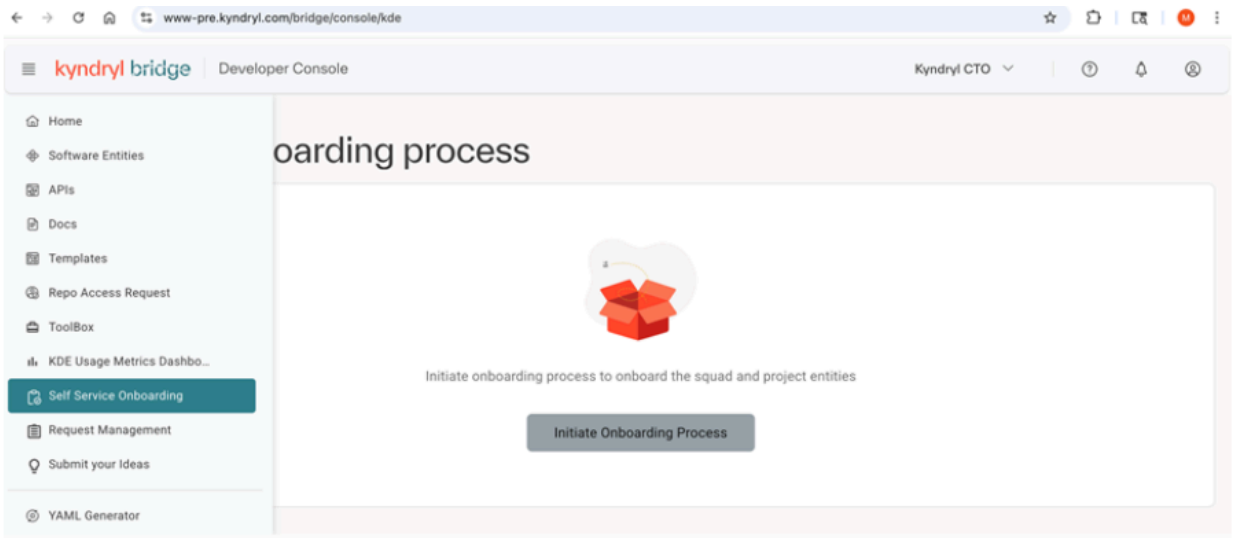
Squad Onboarding

Squad onboarding is the first step in the self-service journey. During this process, a user requests elevation to the Squad Owner role, which grants permissions to manage squad members and technical entities.

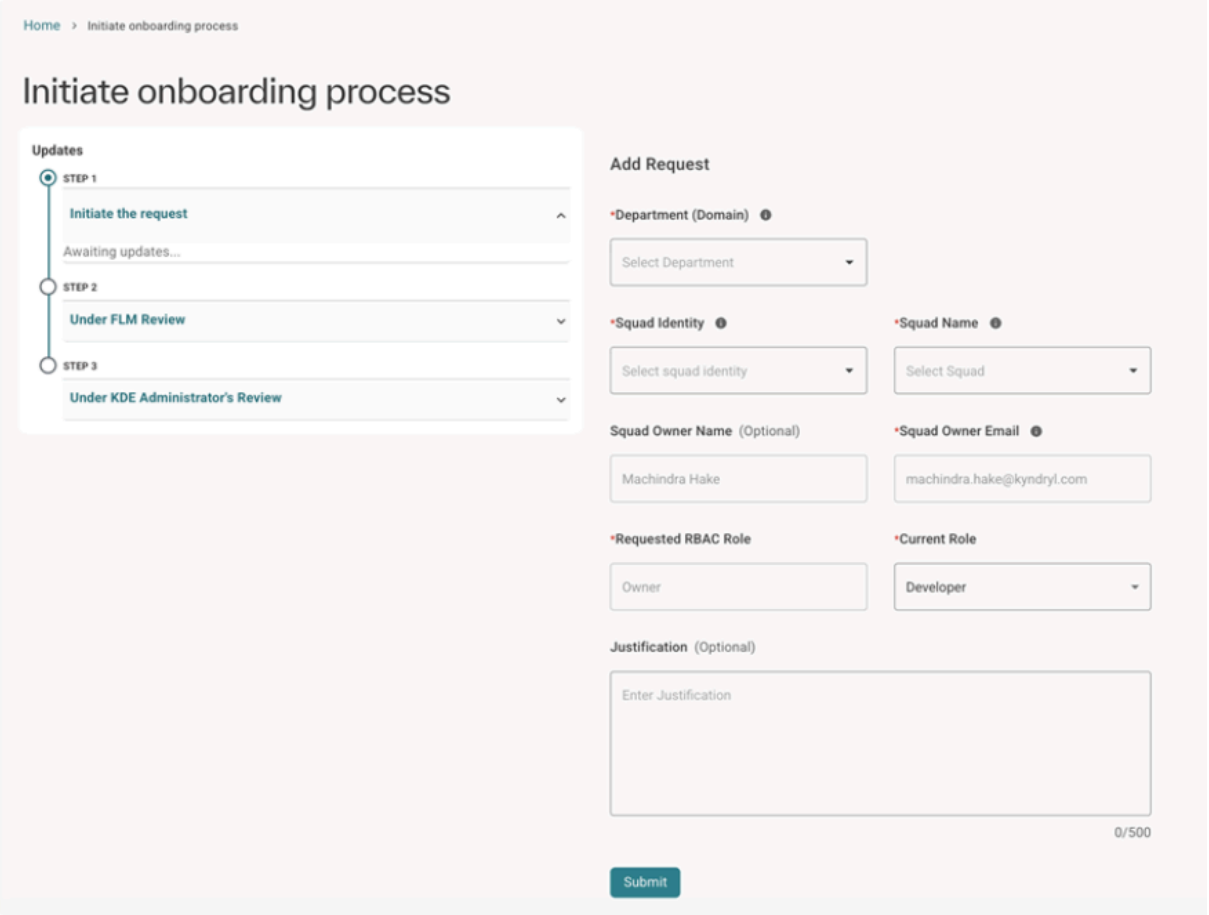
Initiating the Request

Follow the steps below to initiate a squad onboarding request:

1. Navigate to the Self-Service Onboarding page in the KDE Portal.
2. The landing screen displays the option to initiate the onboarding process.
3. Click the "Initiate Onboarding Process" button.



4. Complete the Request Details form:



- Department: Select your organization. This automatically populates the Squad Identity prefix based on GEMU GitHub team mappings.
- Squad Name: Select your squad from the fetched ADO teams list. If not listed, use the "Add new squad" option to manually enter the squad name.
- Squad Owner Email: Auto-populated from your user profile.
- Requested RBAC Role: Auto-filled as Owner.
- Current Role: Select your current role from the available options.

During onboarding, the system automatically provisions role-based access control (RBAC) groups:

- Owner (Admin): Full administrative access for squad, entities, and members.
- Developer (Read-Write): Can create and modify systems and entities.
- Viewer (Read-Only): View-only access to entities.

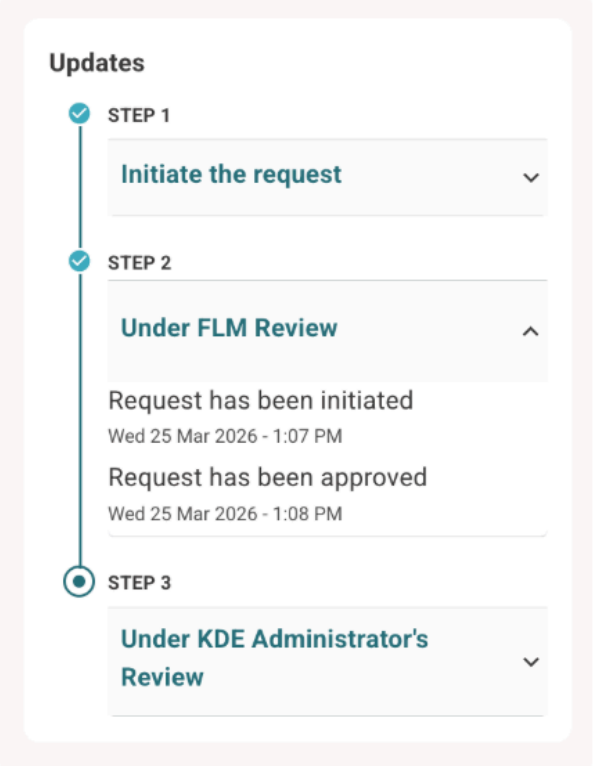
5. Click Submit to send the onboarding request for approval.

Approval Process

After submission, the onboarding request progresses through a transparent approval workflow, which is visualized using the Squad Onboarding Stepper.

The approval stages include:

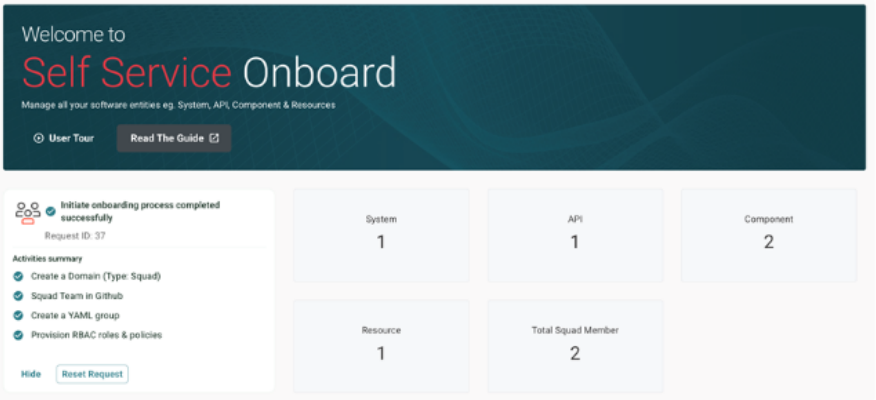
1. Initiate the Request – Request successfully submitted.
2. Under FLM Review – Approval from the First-Line Manager.
3. Under KDE Administrator Review – Final validation by the KDE platform administrators.



Note: The current status of the onboarding request can be tracked at any time from the onboarding landing page.

Squad Management

Once the onboarding request is approved, you are granted Squad Owner access and redirected to the Squad Dashboard. This dashboard provides centralized control over squad members and project entities. And visibility into entity statistics, and onboarding activity status.



Activity Summary

The Activity Summary card confirms successful execution of backend onboarding steps.

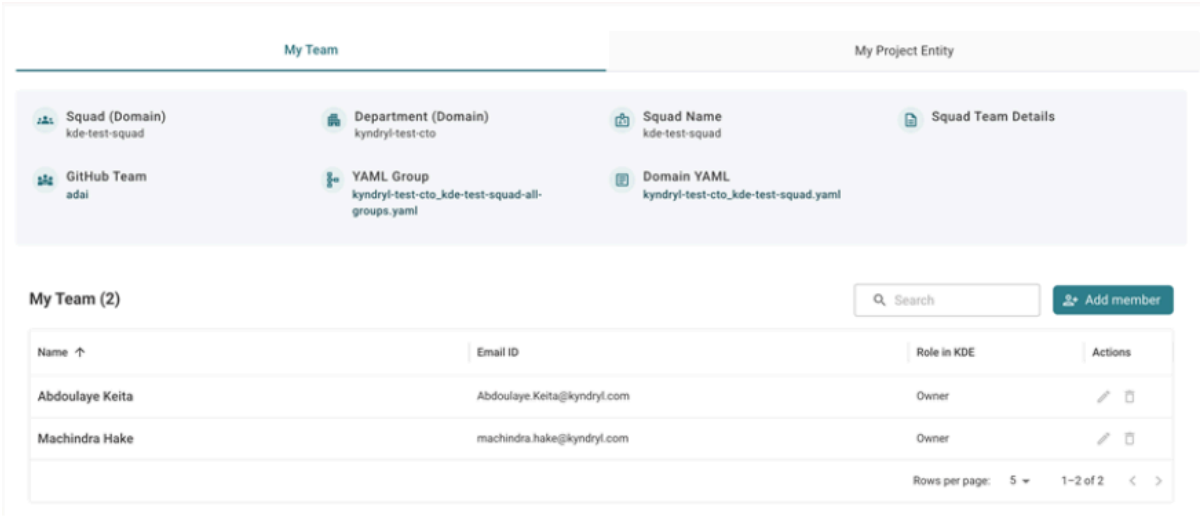
- Domain (Squad) creation
- GitHub squad team setup
- YAML group creation
- RBAC role and policy provisioning

Entity & Squad Statistics

The dashboard displays real-time counts of Systems, APIs, Components, Resources, and Total Squad Members.

My Team Tab

The My Team tab displays squad-level configuration details and member information.



Header Information

The header section displays important squad metadata, including:

- Squad & Department mapping.
- GitHub Team and YAML Group references.
- Domain YAML configuration link.

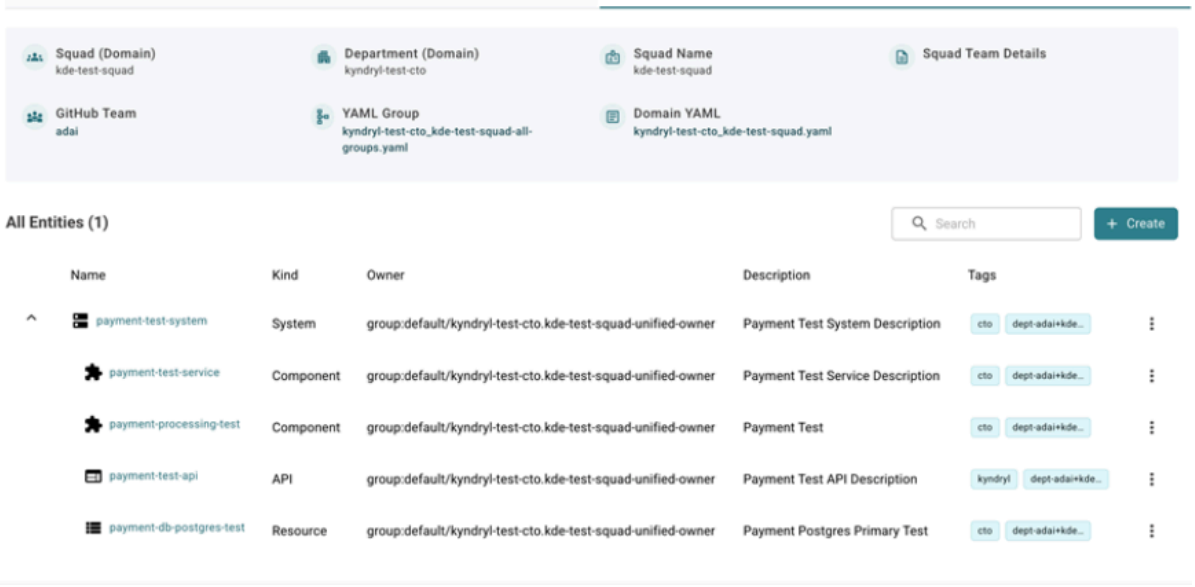
Managing Members

The squad member table allows you to perform the following actions:

- Add Members: Invite new users to the squad.
- Edit Role: Update a member's role using the pencil icon.
- Remove Members: Remove users using the delete (trash) icon.
- Search: Quickly locate members using the search bar.

System & Entity Onboarding

The My Project Entity tab serves as the primary workspace for managing systems and their associated entities. Entities are displayed in a hierarchical structure, with Systems as top-level nodes.



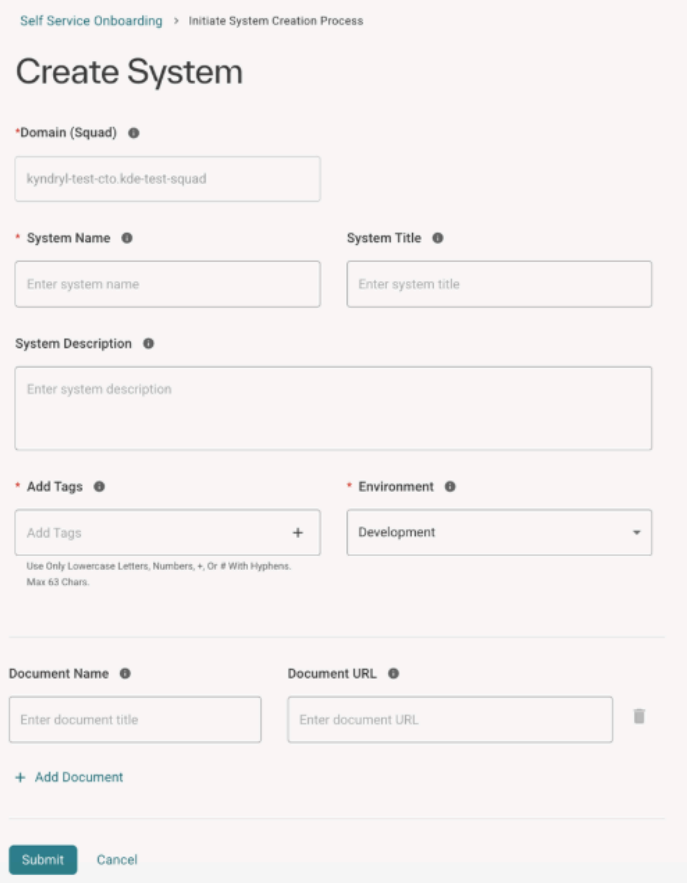
Creating a System

To onboard a new system, perform the following steps:

1. Click the + Create button in the top-right corner of the dashboard.
2. Complete the Create System form:

- Domain (Squad): Auto-filled based on your squad context.
- System Name & Title: Unique identifier and display name.
- Description: Brief explanation of the system's purpose.
- Tags: Mandatory for classification and discoverability.
- Environment: Target environment (e.g., Development, Test, Production).
- Documents: Reference documentation names and URLs.

3. Click Submit to create the system.



Adding Child Entities

APIs, Components, and Resources are created under an existing system to maintain logical grouping.

Steps to add a child entity:

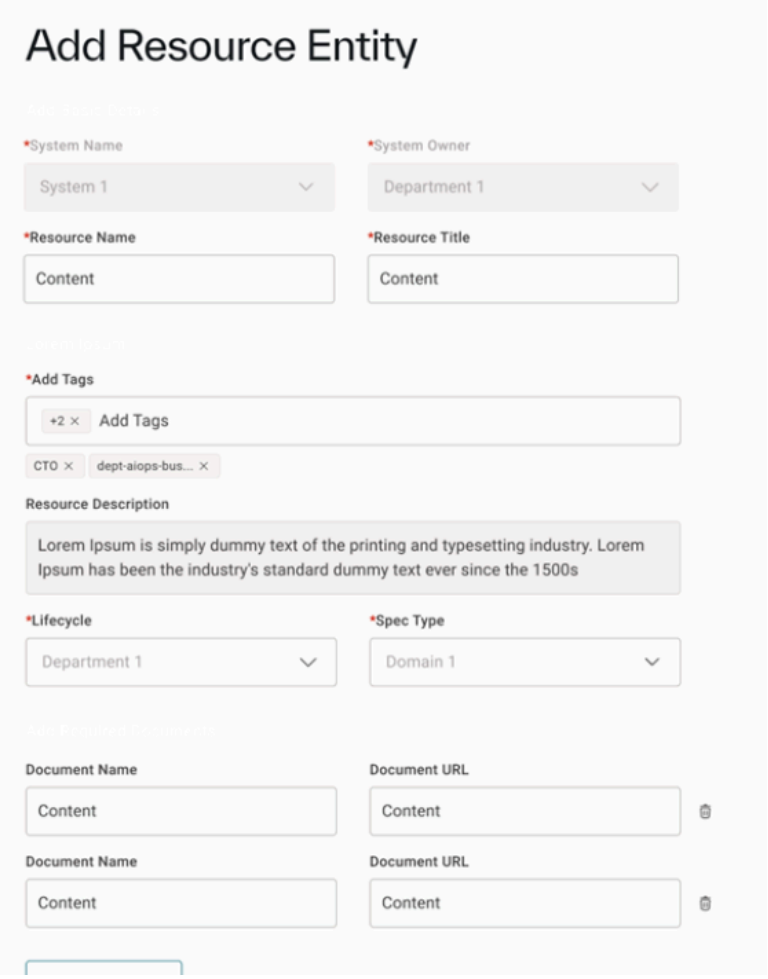
1. Locate the parent system in the All Entities table.
2. Click the three-dot (ellipsis) menu for the system.
3. Select Create API, Create Component, or Create Resource.
4. Complete the entity-specific form, including:

- Basic Details: Name, Title, Description.
- Tags: Mandatory classification tags.
- Technical Details: Repository links, lifecycle, specification type, etc.
- Documents: Related references and URLs.

5. Click Save to create the entity.

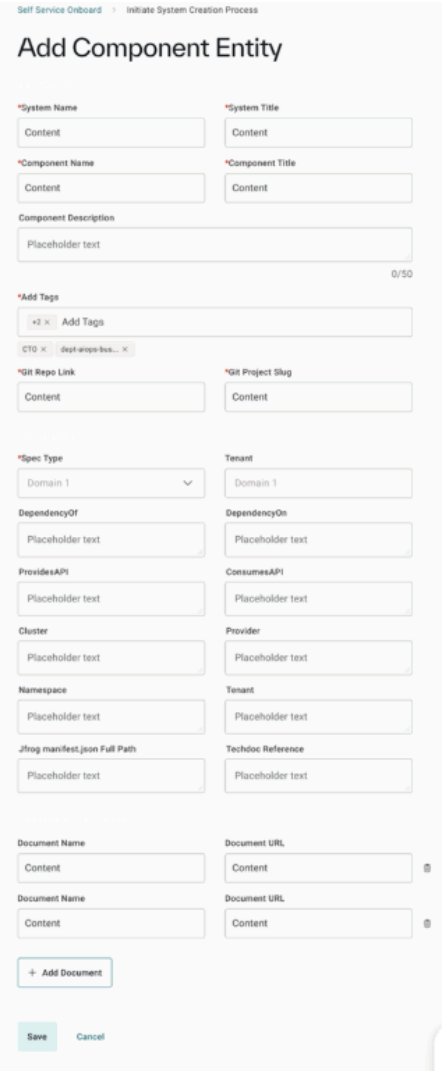
Add Resource Entity

Resource entities represent infrastructure or external dependencies such as databases or messaging systems.



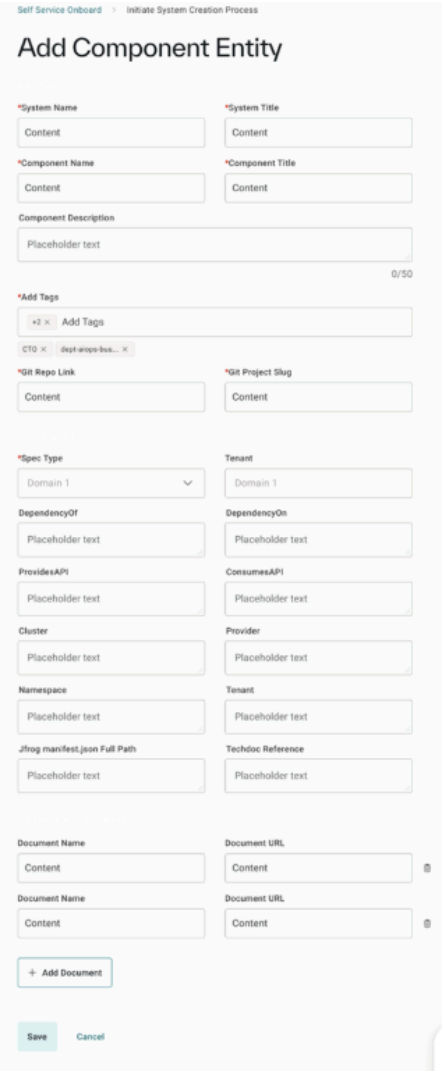
Add Component Entity

Component entities describe deployable services and include dependency mapping, cluster details, and API relationships.



Add API Entity

API entities capture interface definitions and integration details including spec type, lifecycle, repositories, and documentation.

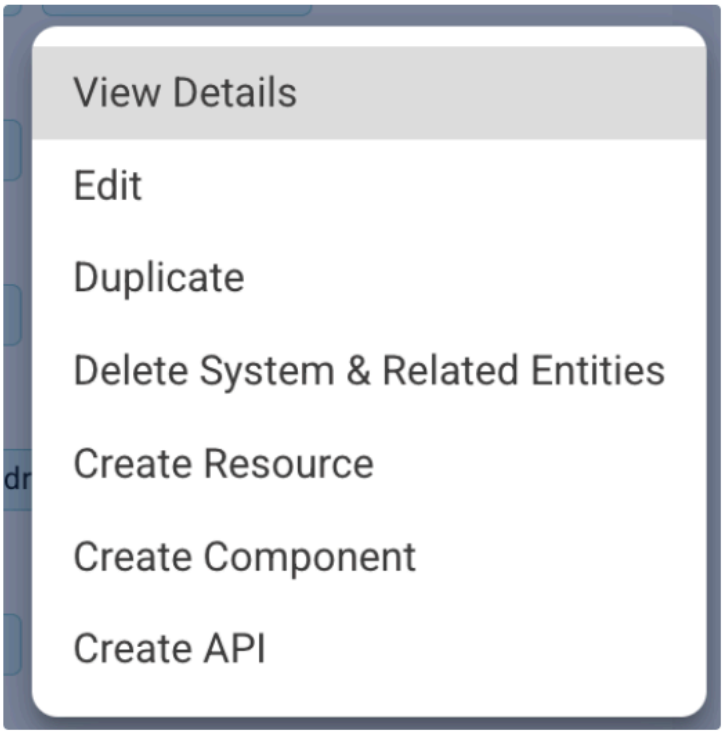


Managing Entities

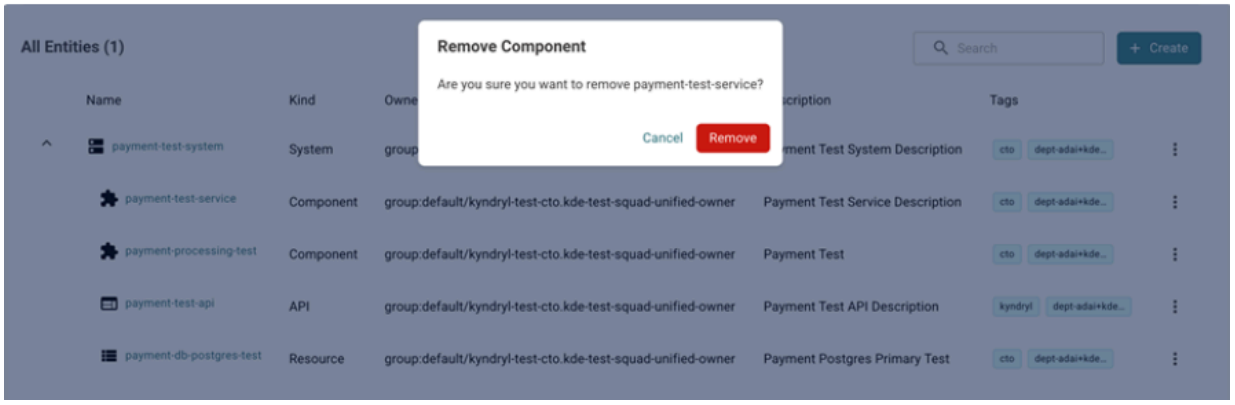
Each entity row provides management actions via the three-dot menu:

- View Details: Open a detailed configuration view.
- Edit: Update editable fields of the entity.
- Duplicate: Create a copy for faster onboarding of similar entities.
- Delete:
  - Systems: Delete system along with all related entities.
  - Child entities: Remove individual APIs, components, or resources.

System entity actions:



Delete the entity:



1.7. KDE VS Code Extension

1. The latest version of VS Code Extension (Version xxxx) offers these capabilities:
2. Ability to view the pipelines, check the status and re-run the pipeline.
3. Ability to view the APIs directly through the Swagger API and execute them.
   Note: If bearer token is available, only then the developer can see API response when executing the Swagger API.

Pre-requisite for Swagger APIs:

1. Share an API definition from the direct Swagger URL without authentication (required to dynamically sync the APIs as the custom extension does not yet have an authentication mechanism built in).
2. For API endpoints that can be accessed only with authentication / authorization, the extension will show or point to the static JSON file of APIs. So, the developer can only execute the APIs from VS Code but the API list will not be dynamically pulled.

Known limitations: Developer Console currently requires a bearer token, as the users of the VS Code custom extension are not authenticated or authorized. Only if bearer token is available, the developer can see API response when executing the Swagger API.

Download the KDE VS Code Extension and Readme file at the end of this section.

🚨 **Note:** Direct clicking is currently not working due to **iframe restrictions** and will be fixed in upcoming releases.

To open the link in a new tab:

- **Desktop Browsers:** Right-click the link and select **“Open link in new tab”** or use **Ctrl/Cmd + Click**.
- **Touchscreen Devices:** Long-press the link and choose **"Open in new tab"**.

Pipeline log visibility

The KDE custom extension for IDEs (VSCode and JetBrains) supports pipeline log visibility, enabling developers to access, search, and manage pipeline job logs directly within their Integrated Development Environment (IDE). This feature enhances troubleshooting, auditing, and monitoring pipeline executions without requiring users to switch between tools.

The main features include:

- **View logs:** Open and review logs related to pipeline executions.
- **Search logs:** Find specific events or errors within the logs.
- **Download and export:** Save logs for further analysis or offline access.
- **Terminal output:** Logs are displayed in the IDE terminal and can be downloaded.
- **Access control:** Currently, users cannot delete the logs to maintain integrity, compliance and retention policies.
- **Clipboard support (optional):** Select, copy, and paste logs into separate files.
- **Pipeline ID display:** Unique identifier for each pipeline log.

The following log fields may be available, depending on IDE limitations:

- Timestamp of each event
- Pipeline Name & Run ID
- Stage/Step Identification
- Status Codes (Success/Error)

- Error Messages & Stack Traces
- Resource Utilization Metrics
- Duration of Processing Steps
- Input/Output Data Counts
- Transformation Details
- User/Service Account that triggered the run
- Environment Variables & Configuration
- Dependencies & Versions

Modularization to the IDE custom extension code for cross-IDE compatibility

The VS Code extension supports integration with JetBrains' IntelliJ IDE, enhancing flexibility, maintainability and efficiency, while simplifying the extension's adaptation to additional development environments.

The key benefits are:

- **Cross-IDE compatibility**: Modular design allows seamless adaptation of the extension to other IDEs with minimal rework.
- **Simplified maintenance:** Bugs and fixes can be addressed in a centralized manner, applying updates across all supported IDE versions.
- **Faster development**: Core functionalities can be reused, reducing the need to rebuild from scratch when adding new IDE support.
- **Consistent user experience:** Modularization ensures uniform behavior across different development environments through shared core components.
- **Reduced testing burden:** Core features require testing only once, with IDE-specific validation focused on integration points.
- **Easier onboarding for contributors:** Clear modular structure simplifies the process for external developers to contribute and expand IDE compatibility.
- **Future-proofing**: Adapting to new IDEs or major version updates becomes easier, requiring changes only in specific adapter modules.

KDE VS Code Extension

Readme File

1.8. Overview

The Kyndryl Development Experience (KDE) page is a centralized portal designed to enhance the developer experience and reduce cognitive load. It is a gateway to discovering and accessing technology assets, technical documents and reusable templates, all in one place. This release is being rolled out selectively to our esteemed CTO teams on a pilot basis. The KDE team will enable the correspondent feature flag and manage access, permissions, and onboarding.

Developers are empowered to view pending tasks, access commonly used developer tools and access resources.

Among all the benefits of using the KDE page, you can find the following:

- Develop or fix code quickly, thanks to ADO tickets and tech document access.
- Collaborate, brainstorm, and share information with the **CTO Tech Forum** Viva Channel.
- Direct access to standard processes and practices for guidelines adherence and onboarding of resources.
- Encourage the adoption of GitHub Copilot.
- Monitor security insights and source code for components and software entities.
- Expedite projects and adopt secure coding standards with starter code configurations and templates.

1.9. Search Capability

The search capability allows to efficiently access to critical information and resources within Kyndryl, leading to better productivity, streamlines workflows, and collaboration. The search process uses an optimized search language that allows you to retrieve multiple search results across all KDE pages. In addition, you can customize the display of search results and view search results.

A search bar is available at the top right of the KDE home page where you can enter from one to 250 characters maximum (including alpha numeric and special characters and multiple words). Once you enter your search keywords, click the **Search** button to execute the search and retrieve the results. A spinning wheel displays every time a search results takes more than five seconds to render.

Information can be searched using filtering and advanced search options based on various criteria (username, teams, docs, APIs name, components, domain, group, application, and templates). If the search is not available, the following message indicating the search had no results will be displayed: “Sorry, no results were found.”

The following items can be searched on KDE:

1. **Components / Service & Application:** Locate microservices, applications, or environments within the internal ecosystem, including service ownership details and configurations, to improve collaboration.
2. **Documentation:** Relevant documentation, API references and guidelines.
3. **Team members:** Look for users by role or name.
4. **Issue and ticket:** Locate existing issues and bugs from the last 5 ADO tickets.
5. **Templates**: Search for templates from the Templates page.

1.10. Software Entities

Software Entities offers developers a comprehensive and centralized platform to view various services and entities. The organized system allows for minimal context switching, centralized access, and clear visibility of elements and their associated details. You can quickly review the owner, components related to each other, associated APIs for this component, or the associated PRs that impact your workflow.

Software Entities not only provides a clear view of existing components within your team and the necessary details to answer your questions, but it also empowers you to create and register components easily and quickly. This feature puts you in control of your development process, allowing you to shape it according to your needs.

When accessing the Developer Console page, select the Software Entities tab.

The following is a list of all software entities:

- API
- Component
- Domain
- Group
- Location
- Resource
- System
- User

Here, it displays the components you have access. Components include:

- Service
- Library
- Resource
- Website
- Documentation
- Other

The Software Entities is designed with your needs in mind, offering flexibility in how you search for elements. You can use the search bar for a quick lookup, or you can filter the list of components by the owner, entity kind, or processing status for a more targeted search. You can also list your favorite components for a quick search.

The entities table display the following details:

- **Name:** Name added to the software entity.
- **System:** Name of the system for the entity.
- **Owner:** Creator of the entity.
- **Type:** Type of component such as service, application, library, resource, website, or documentation.
- **Lifecycle:** Status of the entity such as pilot, production, draft.
- **Description:** A small description of what is this entity.
- **Tags:** Any tags added when creating the software entity.

- **Team or Reviewers:** Users or team designated to review this entity.
- **Actions:** Enables the capability to open the software entity, edit its details and mark it as favorite for quicker access. Favorited entities appear in the “Starred Entities” section of the Home page.

Once the element has been selected, you can view additional details by accessing the overview tab for each item:

- **Insights:** A separate link is available to view the Source Code for component.
- **Documents:** A link to view Tech Docs for each component. If no documentation is linked to this component, a message displays indicating “no tech docs available”.
- **Dependencies:** A list of dependencies between component that allows a user to understand the impact of a change in an upstream component on a component that is downstream.
- **Useful Links:** All links where the component is deployed.
- **Announcement:** Any news concerning component updates that might indirectly affect this component.
- **API:** Here, you can find the APIs exposed by this component as well as the APIs consumed by this component along with their correspondent link to ensure a direct access to the Developer Console API board.
- **Scorecards:** The scorecards tab displays the standards (industry standards or Organization specific standard) and allows to compare your work against the standard. Currently, the GitHub scorecards associated to this component show as per the GitHub best practices along with a adherence statistic to the baseline.

A component can be created from the UI by adding the component from the existing templates or importing from GitHub.

Creating a new component: If you have the correspondent permissions, you can create new software components using standard templates for each kind of component. These templates are pre-defined structures that guide you in providing the necessary details for each type of component, ensuring consistency and completeness in your component creation process.

Registering an existing component: Developers can register the component by entering the URL of the source code repository by linking to an existing entity file or linking to a repository.

Creating a component from API: The option to create the component using API should also be available.

Search function

The search option, located on the left-hand side of the **Software Entities** section, allows users to efficiently narrow down and locate specific entities. You can filter by:

- Type
- Entity Kind
- Tags

Service-level log access in KDE

Kyndryl Developer Experience (KDE) integrates with Environment Management (Container Cluster Management) to offer streamlined access to pod-level service logs and cluster health data. This integration allows developers, with appropriate roles, to quickly access the logs and view necessary insights to take corrective actions, such as addressing memory issues.

Prerequisites:

- Required annotations must be added.
- Cluster details must be onboarded as part of the entity definition.

Accessing logs and cluster health data:

- Select the preferred application.
- Click the **DevOps Foundation** tab
- Select **Environment Management** to view the pod logs and cluster health details.

Benefits:

- **Access to logs:** Developers can view service-level logs easily from KDE.
- **Cluster health insights:** The integration with CCM enables visibility into the cluster's health, providing critical data for troubleshooting.

1.11. Software Extensions

Kyndryl Developer Experience (KDE) allows users to centralize important developer functions by providing access to multiple development environments and support data (technical documentation, microservices, and APIs). Software extensions allow to have different functionalities from other applications/tools and show additional metadata about a software component on the software catalog.

The following is a list of open source plugins:

- GitHub Actions
- GitHub Pull Requests
- GitHub Code Insights
- GitHub Security Insights / CodeQL
- ArgoCD
- Kubernetes
- Kubernetes Topology
- Prometheus
- Grafana
- TechRadar (Base)
- Janus RBAC
- SonarQube
- Dev Tools
- EntityValidator
- Entity Feedback

The following is a list of custom Kyndryl Developed Plugins / Templates:

- GitHub Secrets Scanning
- Jfrog Artifacts
- Tech Insights (Custom) / Scorecards
- Home Page (Custom)
- Azure Devops Ticketing
- Software Templates - NextJS, NodeJS, Spring Boot, Python Flask, Shidoka React,
- Ruby, GoLang

1.12. Toolbox

The Toolbox is designed to streamline your development and design tasks, making them even easier. This section provides a collection of commonly used tools, including encoding, data generation, conversion utilities, and more all in one place for convenience.

All data remains within this domain, ensuring security and preventing unauthorized access.

How to use the Toolbox

From the Developer Console side navigation panel, select ToolBox.

Click on the cards below to select a tool.

Use the left-side navigation to browse available tools.

1.13. Understanding KDE Templates: Creation, Onboarding and Usage

Adrian Doroiman

Associate Director, Software Architecture

TLDR for quick links. More generic information about Backstage templates is in the main body of this page.

Scaffolder Templates

Template repository

Custom actions repository

Custom UI fields repository

Contents

Introduction to KDE Templates

KDE templates are a foundational component within the Backstage developer portal - which KDE uses as a framework, designed to standardize and accelerate the creation of new software components, services, or resources. By leveraging templates, organizations can ensure consistency, enforce best practices, and streamline onboarding for new projects or teams.

A KDE template defines a repeatable process for generating new entities—such as microservices, libraries, or documentation sites—based on predefined configurations and scaffolding logic. This approach not only reduces manual setup time but also helps maintain compliance with architectural and operational standards across the organization.

Templates are typically authored in YAML and can include input parameters, validation rules, and automation steps. When a user initiates a template, Backstage guides them through a form-driven workflow, collects the necessary information, and then executes the template logic to provision the new resource.

Prerequisites for Creating a New KDE Template

Before creating a new KDE template, it is important to ensure that certain prerequisites are met to facilitate a smooth and effective template development process.

- **Access to a KDE Instance:** You must have access to a running KDE instance with the necessary permissions to add or modify templates.
- **Familiarity with YAML:** Backstage templates are typically defined in YAML format. Understanding YAML syntax is essential for authoring and configuring templates.
- **Knowledge of Organizational Standards:** Be aware of Kyndryl's best practices, naming conventions, and compliance requirements to ensure the template aligns with internal guidelines.
- **Source Code Repository:** A GitHub repository is required to store and manage the template files and any associated scaffolding logic.
- **Template Inputs and Outputs:** Clearly define the parameters and expected outputs for the template to ensure it meets the intended use cases.
- **Testing Environment:** Access to a test environment is recommended for validating the template before deploying it to production.

.

Template Structure

When starting development of a new template, the minimum that is required is a template.yaml file. A simple template.yaml definition is shown below

apiVersion: scaffolder.backstage.io/v1beta3

kind: Template

some metadata about the template itself

metadata:

name: v1beta3-demo

title: Test Action template

description: scaffolder v1beta3 template demo

spec:

owner: backstage/techdocs-core

type: service

these are the steps which are rendered in the frontend with the form input

parameters:

- title: Fill in some steps

required:

- name

properties:

name:

```
      title: Name
      type: string
      description: Unique name of the component
      ui:autofocus: true
      ui:options:
        rows: 5
    - title: Choose a location
  required:
    - repoUrl
  properties:
    repoUrl:
      title: Repository Location
      type: string
      ui:field: RepoUrlPicker
      ui:options:
        allowedHosts:
          - github.com

# here are the steps that are executed in series in the scaffolder backend
steps:
  - id: fetch-base
    name: Fetch Base
    action: fetch:template
    input:
      url: ./template
      values:
        name: ${{ parameters.name }}

  - id: fetch-docs
    name: Fetch Docs
    action: fetch:plain
```

```
  input:
    targetPath: ./community
    url: https://github.com/backstage/community/tree/main/backstage-community-sessions

    - id: publish
  name: Publish
  action: publish:github
  input:
    description: This is ${{ parameters.name }}
    repoUrl: ${{ parameters.repoUrl }}
    defaultBranch: 'main'

    - id: register
  name: Register
  action: catalog:register
  input:
    repoContentsUrl: ${{ steps['publish'].output.repoContentsUrl }}
    catalogInfoPath: '/catalog-info.yaml'
```

Onboarding a Template into KDE

Onboarding a template into Backstage involves registering the template so it becomes available for use within the Backstage developer portal. This process ensures that the template is discoverable, properly configured, and ready to be used by teams to scaffold new projects or resources.

- **Template Registration:** Add the template's YAML definition to the KDE catalog, typically by placing the template file in a version-controlled repository and referencing its location in the KDE configuration.
- **Catalog Refresh:** Trigger a catalog refresh or reload so KDE detects and indexes the new template. This step makes the template visible in the Software Templates section of the portal.
- **Template Metadata:** Ensure the template includes all required metadata, such as name, description, owner, and input parameters. This information helps users identify and select the appropriate template for their needs.
- **Validation and Testing:** Before making the template broadly available, test it in a staging or development environment to confirm that it works as intended and aligns with organizational standards.
- **Documentation:** Provide clear documentation and usage instructions, either embedded within the template or as a separate resource, to guide users through the template's workflow and requirements.

Once onboarded, the template can be used by teams to quickly and consistently generate new components, ensuring alignment with best practices and reducing manual setup effort.

Step-by-Step Guide to Using a Newly Onboarded KDE Template

Applying a new KDE Template involves a series of guided steps within the KDE portal. The process is designed to be intuitive and ensures that all necessary information is collected and validated before the template is executed.

- **Access the Component Creation Page:** Navigate to the Software Templates section, typically available at /create in your KDE instance. For local development, this is often http://localhost:3000/create.
- **Select a Template:** Browse the available templates and choose the one that matches your intended use case. Each template may have a unique set of input requirements.
- **Provide Input Variables:** Fill in the required variables as prompted. These may include project name, description, owner, and other parameters specific to the template.
- **Specify Backstage Metadata:** Enter additional fields required for KDE usage, such as the owner
- **Run the Template:** After confirming all inputs, initiate the component creation process. A live progress popup will display the status of each step. If any step fails, you can review logs for troubleshooting.
- **Cancel if Needed:** You can cancel the process at any time. If you do, an abort signal is sent, and subsequent steps will not be executed. The current step will only be cancelled if supported.
- **View the Created Component:** Once the template completes successfully, use the "View in Catalog" button to access the newly registered component in the Backstage catalog.

This workflow ensures that new components are created efficiently, with all necessary metadata and organizational standards applied from the outset.

Best Practices and Troubleshooting Tips

Adhering to best practices when working with Backstage templates ensures a smooth experience for both template authors and users. Additionally, being aware of common troubleshooting steps can help resolve issues quickly and maintain template reliability.

- **Keep Templates Modular:** Design templates to be modular and reusable, allowing teams to adapt them for various use cases without duplicating logic.
- **Document Inputs and Outputs:** Clearly document all required input parameters and expected outputs within the template and supporting documentation to minimize user confusion.
- **Validate Early and Often:** Use validation rules in the template to catch missing or incorrect input values before execution begins.
- **Test in a Development Environment:** Always test new or updated templates in a non-production environment to ensure they function as intended and comply with organizational standards.
- **Version Control:** Store templates in a version-controlled repository to track changes, facilitate collaboration, and enable rollback if issues arise. By convention, you should store each template in its own repository.
- **Monitor Execution Logs:** Encourage users to review execution logs for each template run. Logs provide valuable insights into failures and can help pinpoint the root cause of issues.
- **Provide Troubleshooting Guidance:** Maintain a troubleshooting guide that addresses common errors, such as missing permissions, incorrect repository URLs, or failed automation steps.
- **Engage with the Community:** Participate in the Backstage community forums and discussions to stay updated on best practices, new features, and solutions to common challenges.