



Red Hat Developer Hub 1.8

Interacting with Red Hat Developer Lightspeed for Red Hat Developer Hub

Leverage Artificial Intelligence (AI)-driven expertise of the Red Hat Developer Lightspeed for Red Hat Developer Hub (Developer Lightspeed for RHDH) virtual assistant to help you use Red Hat Developer Hub (RHDH)

Red Hat Developer Hub 1.8 Interacting with Red Hat Developer Lightspeed for Red Hat Developer Hub

Leverage Artificial Intelligence (AI)-driven expertise of the Red Hat Developer Lightspeed for Red Hat Developer Hub (Developer Lightspeed for RHDH) virtual assistant to help you use Red Hat Developer Hub (RHDH)

Legal Notice

Copyright © Red Hat.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

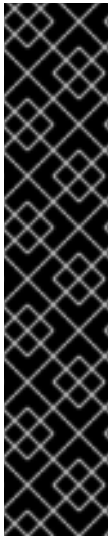
Leverage Artificial Intelligence (AI)-driven expertise of the Red Hat Developer Lightspeed for Red Hat Developer Hub (Developer Lightspeed for RHDH) virtual assistant to help you use Red Hat Developer Hub (RHDH)

Table of Contents

CHAPTER 1. INTERACTING WITH RED HAT DEVELOPER LIGHTSPEED FOR RED HAT DEVELOPER HUB	3
1.1. ABOUT DEVELOPER LIGHTSPEED FOR RHDH	3
1.2. SUPPORTED ARCHITECTURE FOR RED HAT DEVELOPER LIGHTSPEED FOR RED HAT DEVELOPER HUB	4
1.2.1. About Lightspeed Core Service and Llama Stack	4
1.3. RETRIEVAL AUGMENTED GENERATION (RAG) EMBEDDINGS	5
1.4. INSTALLING AND CONFIGURING RED HAT DEVELOPER LIGHTSPEED FOR RED HAT DEVELOPER HUB	5
1.5. CUSTOMIZING DEVELOPER LIGHTSPEED FOR RHDH	14
1.5.1. Gathering feedback in Developer Lightspeed for RHDH	14
1.5.2. Updating the system prompt in Developer Lightspeed for RHDH	15
1.5.3. Customizing the chat history storage in Developer Lightspeed for RHDH	15
1.6. USING DEVELOPER LIGHTSPEED FOR RHDH	16
1.6.1. Using Developer Lightspeed for RHDH to start a chat for the first time	17
1.6.2. Using Developer Lightspeed for RHDH to create new chats after the first chat	17
1.6.3. Using Developer Lightspeed for RHDH to view chat history	18
1.6.4. Using Developer Lightspeed for RHDH to delete a chat	18
1.7. APPENDIX: LLM REQUIREMENTS	18
1.7.1. Large language model (LLM) requirements	18
1.7.2. OpenAI	19
1.7.3. Ollama	19
1.7.4. vLLM	19
1.8. APPENDIX ABOUT USER DATA SECURITY	19
1.8.1. About data use	20
1.8.2. About feedback collection	20
1.8.3. About Bring Your Own Model	20
1.8.4. Your responsibility	20

CHAPTER 1. INTERACTING WITH RED HAT DEVELOPER LIGHTSPEED FOR RED HAT DEVELOPER HUB

1.1. ABOUT DEVELOPER LIGHTSPEED FOR RHDH



IMPORTANT

This section describes Developer Preview features in the Red Hat Developer Lightspeed for Red Hat Developer Hub plugin. Developer Preview features are not supported by Red Hat in any way and are not functionally complete or production-ready. Do not use Developer Preview features for production or business-critical workloads. Developer Preview features provide early access to functionality in advance of possible inclusion in a Red Hat product offering. Customers can use these features to test functionality and provide feedback during the development process. Developer Preview features might not have any documentation, are subject to change or removal at any time, and have received limited testing. Red Hat might provide ways to submit feedback on Developer Preview features without an associated SLA.

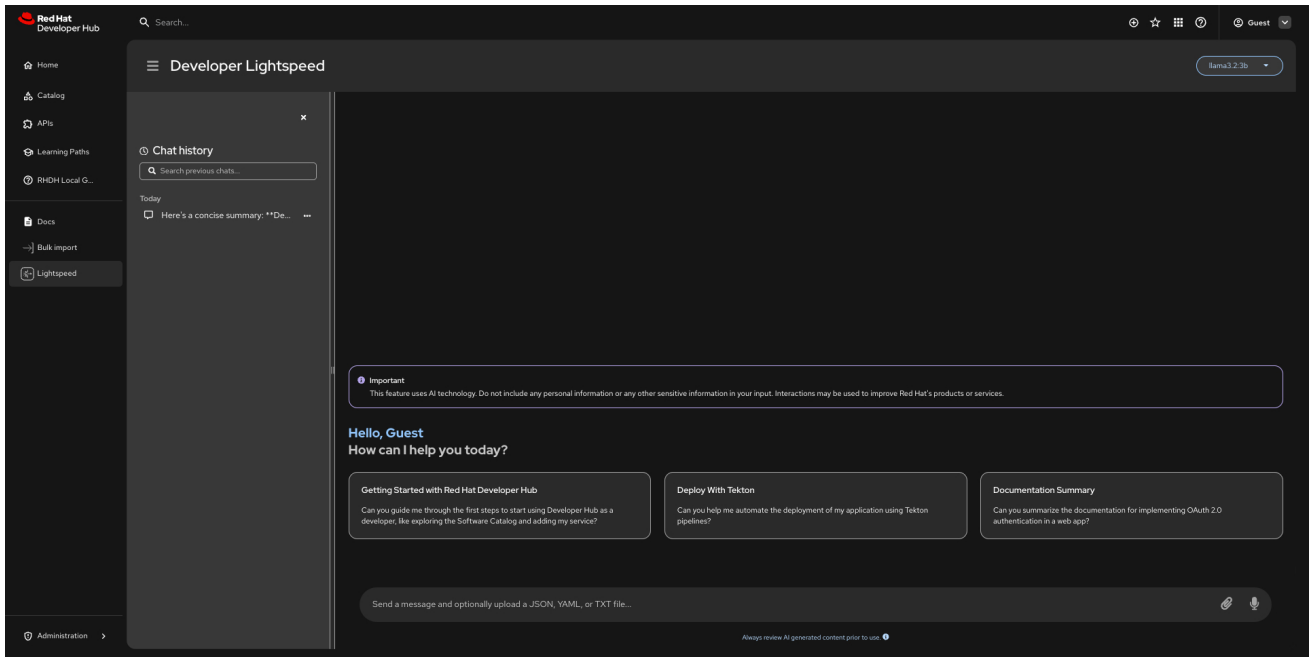
For more information about the support scope of Red Hat Developer Preview features, see [Developer Preview Support Scope](#).

This early access program enables customers to share feedback on the user experience, features, capabilities, and any issues encountered. Your input helps ensure that Developer Lightspeed for RHDH better meets your needs when it is officially released and made generally available.

Red Hat Developer Lightspeed for Red Hat Developer Hub (Developer Lightspeed for RHDH) is a virtual assistant powered by generative Artificial Intelligence (AI) designed for Red Hat Developer Hub(RHDH). The assistant offers in-depth insights into RHDH, including its wide range of capabilities. You can interact with this assistant to explore and learn more about RHDH in greater detail.

Developer Lightspeed for RHDH provides a natural language interface within the RHDH console, helping you easily find information about the product, understand its features, and get answers to your questions.

You can experience Developer Lightspeed for RHDH Developer Preview by installing the Developer Lightspeed for Red Hat Developer Hub plugin within an existing RHDH instance. Alternatively, if you prefer to test it locally first, you can try Developer Lightspeed for RHDH using RHDH Local.



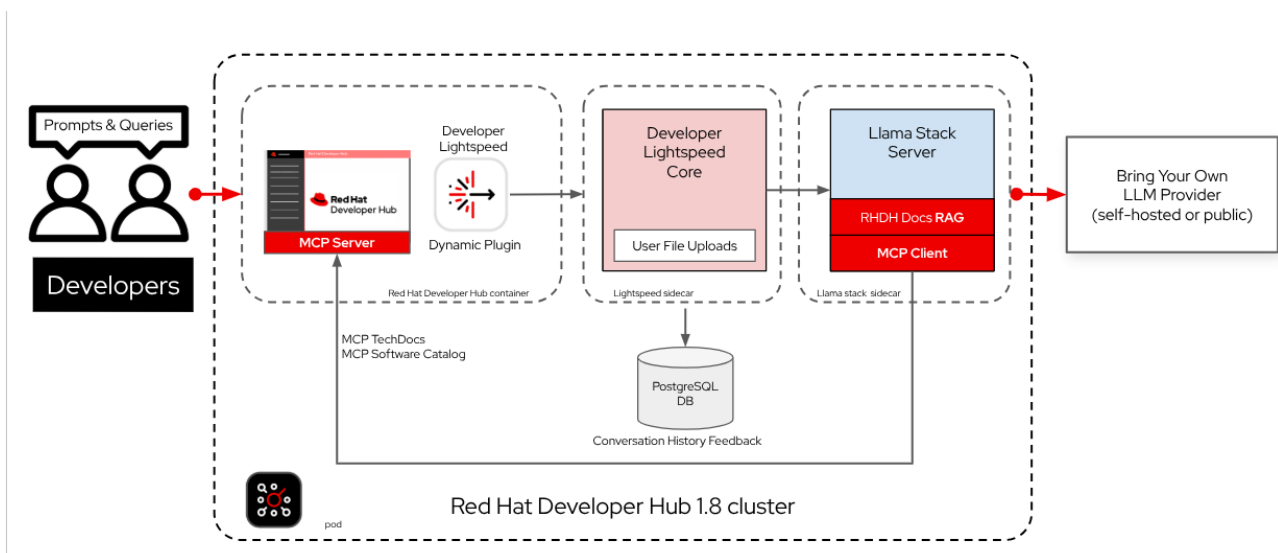
Additional resources

- [RHDH Local](#)

1.2. SUPPORTED ARCHITECTURE FOR RED HAT DEVELOPER LIGHTSPEED FOR RED HAT DEVELOPER HUB

Developer Lightspeed for RHDH is available as a plugin on all platforms that host RHDH. It requires two sidecar containers: the Lightspeed Core Service (LCS) and the Llama Stack service.

The LCS container acts as the intermediary layer, which interfaces with and manages the Llama Stack service.



Additional resources

- [Red Hat Developer Hub Life Cycle and supported platforms](#)

1.2.1. About Lightspeed Core Service and Llama Stack

The Lightspeed Core Service and Llama Stack deploy together as sidecar containers to augment RHDH functionality.

The Llama Stack delivers the augmented functionality by integrating and managing core components, which include:

- Large language model (LLM) inference providers
- Model Context Protocol (MCP) or Retrieval Augmented Generation (RAG) tool runtime providers
- Safety providers
- Vector database settings

The Lightspeed Core Service serves as the Llama Stack service intermediary. It manages the operational configuration and key data, specifically:

- User feedback collection
- MCP server configuration
- Conversation history

Llama Stack provides the inference functionality that LCS uses to process requests. For more information, see [What is Llama Stack](#).

The Red Hat Developer Lightspeed for Red Hat Developer Hub plugin in RHDH sends prompts and receives LLM responses through the LCS sidecar. LCS then uses the Llama Stack sidecar service to perform inference and MCP or RAG tool calling.



NOTE

Red Hat Developer Lightspeed for Red Hat Developer Hub is a Developer Preview release. You must manually deploy the Lightspeed Core Service and Llama Stack sidecar containers, and install the Red Hat Developer Lightspeed for Red Hat Developer Hub plugin on your RHDH instance.

1.3. RETRIEVAL AUGMENTED GENERATION (RAG) EMBEDDINGS

The Red Hat Developer Hub documentation serves as the Retrieval-Augmented Generation (RAG) data source.

RAG initialization occurs through an initialization container, which copies the RAG data to a shared volume. The Llama Stack sidecar then mounts this shared volume to access the RAG data. The Llama Stack service uses the resulting RAG embeddings in the vector database as a reference. This allows the service to provide citations to production documentation during the inference process.

1.4. INSTALLING AND CONFIGURING RED HAT DEVELOPER LIGHTSPEED FOR RED HAT DEVELOPER HUB

Developer Lightspeed for RHDH consists of several components which work together to deliver virtual assistant (chat) functionality to your developers. The following list main components:

Llama stack server (container sidecar)

Based on open source [Llama Stack](#), this service operates as the main gateway to your LLM inferencing provider for chat services. Its modular architecture supports the integration of other services, such as Model Context Protocol (MCP). To support the chat functionality of Developer Lightspeed for RHDH, you must integrate your LLM provider with the Llama Stack server. This dependency on external LLM providers is called *Bring Your Own Model* or BYOM.

Lightspeed Core Service (LCS) (container sidecar)

Based on the open source [Lightspeed Core](#), this service extends the Llama Stack server by providing features such as chat history maintenance and user feedback gathering.

Red Hat Developer Lightspeed for Red Hat Developer Hub (dynamic plugins)

These plugins are required to enable the Developer Lightspeed for RHDH user interface within your RHDH instance.

Configuring these components to initialize correctly and communicate with each other is essential in order to provide Developer Lightspeed for RHDH to your users.

TIP

If you are upgrading from the previous Developer Lightspeed for RHDH Developer Preview with Road-Core Service, you must remove all existing Developer Lightspeed for RHDH configurations and settings before you reinstall.

To prevent or resolve upgrade inconsistencies, first drop and recreate the dynamic plugins volume.

This reinstallation is required due to the following fundamental architectural changes:

- The previous release used Road-Core Service as a sidecar container for interfacing with LLM providers.
- The updated architecture replaces {rcs-short} with the new Lightspeed Core Service and Llama Stack server, which necessitates new configurations for the plugins, volumes, containers, and secrets.

Prerequisites

- You are logged in to your OpenShift Container Platform account.
- You have an RHDH instance installed using either the Operator or the Helm chart.
- You have created a [custom dynamic plugins ConfigMap](#).

Procedure

You must manually install and configure the Developer Lightspeed for RHDH plugin, the Lightspeed Core Service (LCS) sidecar container, and the Llama Stack sidecar container.

1. Create the Lightspeed Core Service (LCS) ConfigMap: The LCS ConfigMap stores the configuration for the Lightspeed Core Service and is mounted to the LCS container.
 - a. In the OpenShift Container Platform web console, navigate to your RHDH instance and select the **ConfigMaps** tab.
 - b. Click **Create ConfigMaps**.
 - c. From the **Create ConfigMap** page, select the **YAML view** option and edit the file using the following structure. This example demonstrates the configuration for the LCS ConfigMap,

typically named **lightspeed-stack**, which connects to the Llama Stack service locally on port **8321**:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: lightspeed-stack
data:
  lightspeed-stack.yaml: |
    name: Lightspeed Core Service (LCS)
    service:
      host: 0.0.0.0
      # Use ${LIGHTSPEED_SERVICE_PORT} if you are not running the Service on port
      '8080'
      # port: ${LIGHTSPEED_SERVICE_PORT}
      auth_enabled: false
      workers: 1
      color_log: true
      access_log: true
      llama_stack:
        use_as_library_client: false
        url: http://localhost:8321
      user_data_collection:
        feedback_enabled: true
        feedback_storage: "/tmp/data/feedback"
        transcripts_enabled: true
        transcripts_storage: "/tmp/data/transcripts"
      authentication:
        module: "noop"
      conversation_cache:
        type: "sqlite"
        sqlite:
          db_path: "/tmp/data/conversations/lcs_cache.db"
```

- d. Click **Create**.
2. Create the Developer Lightspeed for RHDH ConfigMap: Create a dedicated Developer Lightspeed for RHDH ConfigMap (**lightspeed-app-config**) to hold specific plugin configurations.
 - a. In the OpenShift Container Platform web console, navigate to your RHDH instance and select the **ConfigMaps** tab.
 - b. Click **Create ConfigMap**.
 - c. From the **Create ConfigMap** page, select the **YAML view** option and add the following example:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: lightspeed-app-config
  namespace: <__namespace__> # Enter your RHDH instance namespace
data:
  app-config.yaml: |-
    backend:
```

```

csp:
  upgrade-insecure-requests: false
img-src:
  - "self"
  - "data:"
  - https://img.freepik.com
  - https://cdn.dribbble.com
  - https://avatars.githubusercontent.com # This is to load GitHub avatars in the UI
script-src:
  - "self"
  - "unsafe-eval"
  - https://cdn.jsdelivr.net

lightspeed:
  # OPTIONAL: Custom users prompts displayed to users
  # If not provided, the plugin uses built-in default prompts
  prompts:
    - title: `Getting Started with Red Hat Developer Hub`
      message: Can you guide me through the first steps to start using {product-short} as
a developer, like exploring the Software Catalog and adding my service?

  # OPTIONAL: Port for lightspeed service (default: 8080)
  # servicePort: ${LIGHTSPEED_SERVICE_PORT}

  # OPTIONAL: Override default RHDH system prompt
  # systemPrompt: "You are a helpful assistant focused on Red Hat Developer Hub
development."

```

- d. Click **Create**.
3. Create Llama Stack Secret file (**llama-stack-secrets**): This file holds sensitive configuration data for your LLM provider and Llama Stack environment variables.



IMPORTANT

Red Hat Developer Hub 1.8 configures the Llama Stack image to use the vllm provider exclusively. The vllm provider supports the OpenAI API schema but does not support connecting directly to the official OpenAI service (api.openai.com). Do not use the official OpenAI API URL or token with vllm in this release. Attempting this might result in errors or improper responses. Native OpenAI provider support is planned for future releases.

- a. In the OpenShift Container Platform web console, go to **Secrets**.
- b. Click **Create → Key/value secret**
- c. In the **Create key/value secret** page, select the **YAML view** option and add the following example:

```

apiVersion: v1
kind: Secret
metadata:
  name: llama-stack-secrets
type: Opaque
stringData:

```

```
VLLM_URL: ""
VLLM_API_KEY: ""
VLLM_MAX_TOKENS: ""
VLLM_TLS_VERIFY: ""
VALIDATION_PROVIDER: ""
VALIDATION_MODEL_NAME: ""
```

where:

VLLM_URL

Required for remote services: Set this to the API endpoint URL of your preferred LLM provider, if it is compatible with the OpenAI API specification (Examples: OpenAI, Red Hat OpenShift AI, vLLM)

VLLM_API_KEY

Required for remote services: Set this to the API key or token required for authentication with your remote LLM provider, if it is compatible with the OpenAI API specification

VLLM_MAX_TOKENS

Optional

VLLM_TLS_VERIFY

Optional

VALIDATION_PROVIDER

Set this as **vllm**, **ollama**, **openai**, depending on the key you have set in this configuration file

VALIDATION_MODEL_NAME

Set the name of the model you want to use for validation

d. Click **Create**.

4. Update the dynamic plugins ConfigMap: Add the Developer Lightspeed for RHDH plugin image to your existing dynamic plugins ConfigMap (**dynamic-plugins-rhdh**).

```
includes:
- dynamic-plugins.default.yaml
plugins:
- package: oci://ghcr.io/redhat-developer/rhdh-plugin-export-overlays/red-hat-developer-hub-backstage-plugin-lightspeed:bs_1.42.5__1.0.3!red-hat-developer-hub-backstage-plugin-lightspeed
  disabled: false
  pluginConfig:
    lightspeed:
      # OPTIONAL: Custom users prompts displayed to users
      prompts:
        - title: 'Getting Started with Red Hat Developer Hub'
          message: Can you guide me through the first steps to start using Developer Hub as a developer, like exploring the Software Catalog and adding my service?
    dynamicPlugins:
      frontend:
        red-hat-developer-hub.backstage-plugin-lightspeed:
          translationResources:
            - importName: lightspeedTranslations
              module: Alpha
```

```

    ref: lightspeedTranslationRef
  applcons:
    - name: LightspeedIcon
      module: LightspeedPlugin
      importName: LightspeedIcon
  dynamicRoutes:
    - path: /lightspeed
      importName: LightspeedPage
      module: LightspeedPlugin
      menuItem:
        icon: LightspeedIcon
        text: Lightspeed
  - package: oci://ghcr.io/redhat-developer/rhdh-plugin-export-overlays/red-hat-developer-hub-
    backstage-plugin-lightspeed-backend:bs_1.42.5__1.0.3!red-hat-developer-hub-backstage-
    plugin-lightspeed-backend
    disabled: false

```

5. Update your deployment configuration: Update the deployment configuration based on how your RHDH instance was installed. You must add two sidecar containers: **llama-stack** and **lightspeed-core**.

- For an Operator-installed RHDH instance (Update Backstage Custom Resource (CR)):
 - i. In the **spec.application.appConfig.configMaps** section of your Backstage CR, add the Developer Lightspeed for RHDH custom app configuration:

```

  appConfig:
    configMaps:
      - name: lightspeed-app-config

```

- ii. Update the **spec.deployment.patch.spec.template.spec.volumes** specification to include volumes for LCS configuration (**lightspeed-stack**), shared storage for feedback (**shared-storage**), and RAG data (**rag-data-volume**):

```

  volumes:
    - configMap:
        name: lightspeed-stack
      name: lightspeed-stack
    - emptyDir: {}
      name: shared-storage
    - emptyDir: {}
      name: rag-data-volume

```

- iii. Add the **initContainers** section to initialize RAG data:

```

  initContainers:
    - name: init-rag-data
      image: 'quay.io/redhat-ai-dev/rag-content:release-1.8-lcs'
      command:
        - "sh"
        - "-c"
        - "echo 'Copying RAG data...'; cp -r /rag/vector_db/rhdh_product_docs /data/
        && cp -r /rag/embeddings_model /data/ && echo 'Copy complete.'"

```

```

volumeMounts:
  - mountPath: /data
    name: rag-data-volume

```

- iv. Add the Llama Stack and the LCS containers to the **spec.deployment.patch.spec.template.spec.containers** section:

```

spec:
  application:
    - extraEnvs:
        secrets:
          - name: lightspeed-secrets
  containers:
    # ... Your existing RHDH container definition ...
    - envFrom:
        - secretRef:
            name: llama-stack-secrets
      image: 'quay.io/redhat-ai-dev/llama-stack:0.1.1' # Llama Stack image
      name: llama-stack
      volumeMounts:
        - mountPath: /app-root/.llama
          name: shared-storage
        - mountPath: /app-root/embeddings_model
          name: rag-data-volume
          subPath: embeddings_model
        - mountPath: /app-root/vector_db/rhdh_product_docs
          name: rag-data-volume
          subPath: rhdh_product_docs
      - image: 'quay.io/lightspeed-core/lightspeed-stack:dev-20251021-ee9f08f' #
        Lightspeed Core Service image
        name: lightspeed-core
        volumeMounts:
          - mountPath: /app-root/lightspeed-stack.yaml
            name: lightspeed-stack
            subPath: lightspeed-stack.yaml
          - mountPath: /tmp/data/feedback
            name: shared-storage
          - mountPath: /tmp/data/transcripts
            name: shared-storage
          - mountPath: /tmp/data/conversations
            name: shared-storage

```

- v. Click **Save**. The Pods are automatically restarted.

- For a Helm-installed RHDH instance (Update the Helm chart):
 - i. Add your dynamic plugins configuration in the **global.dynamic** property.
 - ii. Add your Developer Lightspeed for RHDH custom app config file to **extraAppConfig**:

```

extraAppConfig:
  - configMapRef: lightspeed-app-config
    filename: app-config.yaml

```

- iii. Add the Llama Stack Secret file to **extraEnvVarsSecrets**:

```
extraEnvVarsSecrets:
  - llama-stack-secrets
```

- iv. Update the **extraVolumes** section to include the LCS ConfigMap (**lightspeed-stack**), shared storage, and RAG data volume:

```
extraVolumes:
  - configMap:
      name: lightspeed-stack
  - emptyDir: {}
      name: shared-storage
  - emptyDir: {}
      name: rag-data-volume
```

- v. Update the **initContainers** section (if supported by your Helm chart structure) to initialize RAG data.

```
initContainers:
  - name: init-rag-data
    image: 'quay.io/redhat-ai-dev/rag-content:release-1.8-lcs'
    command:
      - "sh"
      - "-c"
      - "echo 'Copying RAG data...'; cp -r /rag/vector_db/rhdh_product_docs /data/
        && cp -r /rag/embeddings_model /data/ && echo 'Copy complete.'"
    volumeMounts:
      - mountPath: /data
        name: rag-data-volume
```

- vi. Add the Llama Stack and LCS container definitions to **extraContainers**



NOTE

If you have Road-Core Service installed from the previous Red Hat Developer Lightspeed for Red Hat Developer Hub configuration, you must replace the older single container configuration found in source with the two sidecars.

```
extraContainers:
  # Llama Stack Container
  - envFrom:
      - secretRef:
          name: llama-stack-secrets
    image: 'quay.io/redhat-ai-dev/llama-stack:0.1.1'
    name: llama-stack
    volumeMounts:
      - mountPath: /app-root/.llama
        name: shared-storage
      - mountPath: /app-root/embeddings_model
        name: rag-data-volume
        subPath: embeddings_model
      - mountPath: /app-root/vector_db/rhdh_product_docs
```

```

      name: rag-data-volume
      subPath: rhdh_product_docs
    # Lightspeed Core Service Container
    - image: 'quay.io/lightspeed-core/lightspeed-stack:dev-20251021-ee9f08f'
      name: lightspeed-core
      volumeMounts:
        - mountPath: /app-root/lightspeed-stack.yaml
          name: lightspeed-stack
          subPath: lightspeed-stack.yaml
        - mountPath: /tmp/data/feedback
          name: shared-storage
        - mountPath: /tmp/data/transcripts
          name: shared-storage
        - mountPath: /tmp/data/conversations
          name: shared-storage

```

vii. Click **Save** and then Helm upgrade.

6. Optional: Manage authorization (RBAC): If you have users who are not administrators, you must [define permissions and roles](#) for them to use Developer Lightspeed for RHDH. The Lightspeed Backend plugin uses Backstage RBAC for authorization.

- For an Operator-installed RHDH instance:
 - i. Configure the required RBAC permission by defining an **rbac-policies.csv** file, including **lightspeed.chat.read**, **lightspeed.chat.create**, and **lightspeed.chat.delete** permissions:

```

p, role:default/_<your_team>_, lightspeed.chat.read, read, allow
p, role:default/_<your_team>_, lightspeed.chat.create, create, allow
p, role:default/_<your_team>_, lightspeed.chat.delete, delete, allow
g, user:default/_<your_user>_, role:default/_<your_team>_

```

- ii. Upload your **rbac-policies.csv** file to an **rbac-policies** ConfigMap in your OpenShift Container Platform project containing RHDH and update your Backstage CR:

```

apiVersion: rhdh.redhat.com/v1alpha3
kind: Backstage
spec:
  application:
    extraFiles:
      mountPath: /opt/app-root/src
    configMaps:
      - name: rbac-policies

```

- For a Helm-installed RHDH instance:
 - i. Configure the required RBAC permission by defining an **rbac-policies.csv** file:

```

p, role:default/_<your_team>_, lightspeed.chat.read, read, allow
p, role:default/_<your_team>_, lightspeed.chat.create, create, allow
p, role:default/_<your_team>_, lightspeed.chat.delete, delete, allow
g, user:default/_<your_user>_, role:default/_<your_team>_

```

- ii. Optional: Declare policy administrators by editing your custom RHDH ConfigMap (**app-config.yaml**) and adding the following code to enable selected authenticated users to configure RBAC policies through the REST API or Web UI:

```

permission:
  enabled: true
rbac:
  policies-csv-file: /opt/app-root/src/rbac-policies.csv
  policyFileReload: true
admin:
  users:
    - name: user:default/<your_policy_administrator_name>

```

Verification

1. Log in to your RHDH instance.
2. In your RHDH navigation menu, you are able to see and access the **Lightspeed** menu item. Clicking this menu item takes you to the Developer Lightspeed for RHDH screen.

1.5. CUSTOMIZING DEVELOPER LIGHTSPEED FOR RHDH

You can customize Developer Lightspeed for RHDH functionalities such as gathering feedback, storing chat history in PostgreSQL, and [configuring Model Context Protocol \(MCP\) tools](#).

1.5.1. Gathering feedback in Developer Lightspeed for RHDH

Feedback collection is an optional feature configured on the LCS. This feature gathers user feedback by providing thumbs-up/down ratings and text comments directly from the chat window.

LCS collects the feedback, the user's query, and the response of the model, storing the data as a JSON file on the local file system of the Pod. A platform administrator must later collect and analyze this data to assess model performance and improve the user experience.

The collected data resides in the cluster where RHDH and LCS are deployed, making it accessible only to platform administrators for that cluster. For data removal, users must request this action from their platform administrator, as Red Hat neither collects nor accesses this data.

Procedure

- To enable feedback collection, in the LCS configuration file (**lightspeed-stack.yaml**), add the following settings:

```

user_data_collection:
  feedback_enabled: true
  feedback_storage: "/tmp/data/feedback"
  transcripts_enabled: true
  transcripts_storage: "/tmp/data/transcripts"

```

- To disable feedback collection, in the LCS configuration file (**lightspeed-stack.yaml**), add the following settings:

```

user_data_collection:
  feedback_enabled: false

```

```
feedback_storage: "/tmp/data/feedback"
transcripts_enabled: true
transcripts_storage: "/tmp/data/transcripts"
```

1.5.2. Updating the system prompt in Developer Lightspeed for RHDH

You can override the default system prompt that Developer Lightspeed for RHDH uses to better frame queries to your LLM. Customizing the system prompt allows you to refine the context, personality, and instructions that the LLM receives, improving the relevance and accuracy of the responses it creates for your specific environment.

Procedure

- To set a custom system prompt, in your Developer Lightspeed for RHDH app config file, add or modify the **lightspeed.systemPrompt** key and set its value to your preferred prompt string as shown in the following example:

```
lightspeed:
  # ... other lightspeed configurations
  systemPrompt: "You are a helpful assistant focused on Red Hat Developer Hub
  development."
```

Set **systemPrompt** to prefix all queries sent by Developer Lightspeed for RHDH to the LLM with this instruction, guiding the model to generate more tailored responses.

1.5.3. Customizing the chat history storage in Developer Lightspeed for RHDH

By default, the Developer Lightspeed for RHDH service stores chat history in a non-persistent local SQL database within in the LCS container. This means that chat history is lost if you create and use a new LCS sidecar. You can manually configure Developer Lightspeed for RHDH to store the chat history persistently as a long-term backup with PostgreSQL by updating your LCS service configuration.

+



WARNING

Configuring Developer Lightspeed for RHDH to use PostgreSQL records prompts and responses, which platform administrators can review. You must assess any data privacy and security implications if user chat history contains private, sensitive, or confidential information. For users that wish to have their chat data removed, they must request their respective platform administrator to perform this action. Red Hat does not collect or access this chat history data.

Procedure

- Configure the chat history storage type in the LCS configuration file (**lightspeed-stack.yaml**) using any of the relevant options:
 - To enable persistent storage with PostgreSQL, add the following configuration:

■

```
conversation_cache:
  type: postgres
  postgres:
    host: _<your_database_host>_
    port: _<your_database_port>_
    db: _<your_database_name>_
    user: _<your_user_name>_
    password: _<postgres_password>_
```

- To retain the default, non-persistent SQLite storage, make sure the configuration is set as shown in the following example:

```
conversation_cache:
  type: "sqlite"
  sqlite:
    db_path: "/tmp/data/conversations/lcs_cache.db"
```

2. Restart your LCS service to apply the new configuration.

1.6. USING DEVELOPER LIGHTSPEED FOR RHDH

Red Hat Developer Lightspeed for Red Hat Developer Hub is designed to support you when performing various tasks during your development workflow.



NOTE

The **Question Validation** feature is enabled by default if you are using the **quay.io/redhat-ai-dev/llama-stack** image without overriding the **run.yaml** configuration file in the image. To disable **Question Validation**, you must mount a **run.yaml** file to the container with the following sections removed:

- **Safety**
- **Shields**
- **External_providers_dir** set to **null**

With **Question Validation** enabled, you can ask Developer Lightspeed for RHDH the following types of questions:

- "Tell me about Red Hat Developer Hub."
- "What are the benefits of RHDH?"
- "Can I use RHDH on an OpenShift Container Platform?"
- "How do I install plugins on Red Hat Developer Hub?"

With **Question Validation** disabled, the scope of prompts you can put to Developer Lightspeed for RHDH is much broader. This allows Developer Lightspeed for RHDH to support you in a much more varied range of work situations as described in the following examples:

- "Analyze this log for me..."

- “Suggest libraries and frameworks I can use to build Event Driven Architecture microservices.”
- “I’m not familiar with this language, so explain to me what this code snippet is doing...”
- “Create a Kubernetes deployment for this service...”
- “Create a test plan for the following scenarios and conditions...”
- “Create a Jira record that describes the following feature...”
- “Draft the end-user documentation describing how to use the following cli command...”

1.6.1. Using Developer Lightspeed for RHDH to start a chat for the first time

You can start a chat with Developer Lightspeed for RHDH for quick answers on a number of topics depending on your settings. You can manually start a chat with the Developer Lightspeed for RHDH or use the following sample prompts we have provided to help you get started:

- **Getting Started with Red Hat Developer Hub**
- **Deploy with Tekton**
- **Create an OpenShift Deployment**

Prerequisites

- You have the Developer Lightspeed for RHDH plugin configured in your RHDH instance.

Procedure

1. In your RHDH navigation menu, click **Lightspeed**.
2. You can start a chat in either of the following ways:
 - To manually start a chat, in the **Send a message** text box, you can do any of the following tasks:
 - Type your query and press **Enter**.
 - To attach a file in the chat, click the **Attach** icon or drag and drop the file in your chat.



NOTE

The following file types are supported: **yaml**, **json**, and **txt**.

- Click **Open**.
- To start a chat using the existing prompts, in the Developer Lightspeed for RHDH virtual assistant interface, click any of the relevant prompt tiles.

1.6.2. Using Developer Lightspeed for RHDH to create new chats after the first chat

After you have started an initial chat with the Developer Lightspeed for RHDH, you can begin a chat on a new topic at any time. Even if you log out and log back in, your previous chats are still available in your chat history for you to view.

Prerequisites

- You have the Developer Lightspeed for RHDH plugin configured in your RHDH instance.

Procedure

1. In your RHDH navigation menu, click **Lightspeed**.
2. In the Developer Lightspeed for RHDH virtual assistant interface, click **New chat**.

1.6.3. Using Developer Lightspeed for RHDH to view chat history

Your chats with Developer Lightspeed for RHDH are automatically saved in your RHDH instance. You can easily revisit your chat history at any time, switch between chats, and revisit any previous chats. Each chat remains active, enabling you to go back to any of your previous chats and continue from where you left off.

Prerequisites

- You have the Developer Lightspeed for RHDH plugin configured in your RHDH instance.

Procedure

1. In your RHDH navigation menu, click **Lightspeed**. Developer Lightspeed for RHDH opens with your previous chat.
2. In the Developer Lightspeed for RHDH virtual assistant interface, do any of the following tasks:
 - Select a chat title to open and view the full chat.
 - In **Search previous chats...**, enter the text that you want to find from the earlier chats.

1.6.4. Using Developer Lightspeed for RHDH to delete a chat

Prerequisites

- You have the Developer Lightspeed for RHDH plugin configured in your RHDH instance.

Procedure

1. In your RHDH navigation menu, click **Lightspeed**. Developer Lightspeed for RHDH opens with your previous chat.
2. In Developer Lightspeed for RHDH, select the vertical ellipsis for the chat title of the chat that you want to delete.
3. Select **Delete**.

1.7. APPENDIX: LLM REQUIREMENTS

1.7.1. Large language model (LLM) requirements

Developer Lightspeed for RHDH follows a *Bring Your Own Model* approach. This model means that to function, Developer Lightspeed for RHDH requires access to a large language model (LLM) which you

must provide. An LLM is a type of generative AI that interprets natural language and generates human-like text or audio responses. When an LLM is used as a virtual assistant, the LLM can interpret questions and provide answers in a conversational manner.

LLMs are usually provided by a service or server. Because Developer Lightspeed for RHDH does not provide an LLM for you, you must configure your preferred LLM provider during installation. You can configure the underlying Llama Stack server to integrate with a number of LLM **providers** that offer compatibility with the OpenAI API including the following inference providers:

- OpenAI (cloud-based inference service)
- Red Hat OpenShift AI (enterprise model builder and inference server)
- Red Hat Enterprise Linux AI (enterprise inference server)
- Ollama (popular desktop inference server)
- vLLM (popular enterprise inference server)

1.7.2. OpenAI

OpenAI offers a range of generative AI models, such as GPT 5, which can be used to provide inference services for applications like Developer Lightspeed for RHDH.

To use OpenAI with Developer Lightspeed for RHDH, you need access to the OpenAI [API platform](#). For more information, see the [OpenAI developer platform documentation](#).

1.7.3. Ollama

Ollama is a powerful and easy-to-use open-source project that simplifies the process of running large language models (LLMs) locally on your computer. It provides a simple command-line interface for downloading, managing, and running a wide variety of open-source models, such as Llama 3, Mistral, and many others, all without requiring a dedicated server or cloud service. By abstracting away the complex setup and dependencies, Ollama makes it accessible for developers, researchers, and enthusiasts to experiment with, build on, and integrate state-of-the-art LLMs into their applications directly from their personal machines.

The open source Ollama server in container form provides a convenient local testbed for LLM models that is very accessible and easily controlled.

Additional resources

- [Ollama](#)
- [Ollama server container](#)

1.7.4. vLLM

[vLLM](#) is an open-source, high-throughput serving engine for large language models (LLMs) that significantly improves upon traditional serving systems. It achieves this by introducing several key optimizations to reduce memory usage and eliminate redundant computations. vLLM prominently increases the number of concurrent requests an LLM can handle, making it a powerful tool for deploying and scaling LLM-based applications.

1.8. APPENDIX ABOUT USER DATA SECURITY

1.8.1. About data use

Developer Lightspeed for RHDH is a virtual assistant you interact with using natural language. Using the Developer Lightspeed for RHDH interface, you send chat messages that Developer Lightspeed for RHDH transforms and sends to the large language model (LLM) provider you have configured for your environment. These messages could potentially contain information provided by your users about themselves, your cluster, cluster resources, or other aspects of your business or working environment.

Developer Lightspeed for RHDH has limited capabilities to filter or redact the information you provide to the LLM. Do not enter information into Developer Lightspeed for RHDH that you do not want to send to the LLM provider. To remind end users not to share private or confidential information, Developer Lightspeed for RHDH begins each new chat with an 'Important' message asking them not to “include personal or sensitive information” in their chat messages.

1.8.2. About feedback collection

Developer Lightspeed for RHDH collects feedback from users who engage with the feedback feature in the virtual assistant interface. If a user submits feedback, the feedback score (thumbs up or down), text feedback (if entered), the user query, and the LLM provider response are stored locally in the file system of the Pod. Red Hat does not have access to the collected feedback data.

1.8.3. About Bring Your Own Model

Developer Lightspeed for RHDH does not provide its own inference services, but uses a *Bring Your Own Model* approach. This means that you can configure the Lightspeed Core Service to talk to the inference server or service of your choice. This also means that you are responsible for ensuring that the configured service meets your particular company policies and legal requirements, including any applicable terms with the third-party model provider. The only technical requirements for inference services are:

- The service must conform to the OpenAI API specification.
- The service must be configured correctly following the installation and configuration instructions. There are many commercial and open source inference services that support the OpenAI API specification for chat completions. The cost, performance, and security of these services can differ and it is up to you to choose, through evaluation and testing, the inference service that best meets your company’s needs.

Additional resources

- [OpenAI API specification](#)

1.8.4. Your responsibility

All of the information your users share in their questions and responses with Developer Lightspeed for RHDH are shared with the LLM inference service you configured. You are responsible for ensuring compliance with your company’s policies regarding the sharing of data with your chosen inference service.