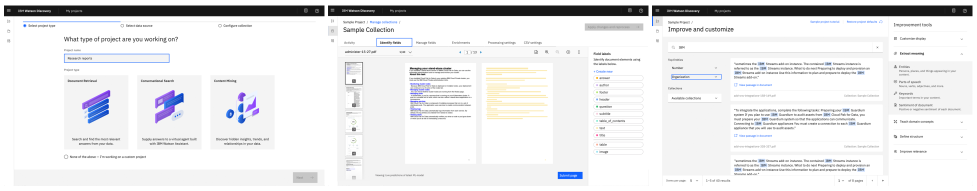
Watson Discovery on Cloud Pak for Data

Version: 4.7.1 Premium IBM



Description

IBM Watson® Discovery for IBM Cloud Pak® for Data is an award-winning AI-powered intelligent search and text-analytics platform that helps you find valuable information that is buried in your enterprise data. Discovery uses innovative, market-leading natural language processing to uncover meaningful insights from complex business documents.

With Discovery, you can:

- Leverage advanced search capabilities, such as table retrieval.
- Use the built-in contract understanding function to search and interpret legal contracts.
- Deploy the full-featured Content Mining application to conduct in-depth analysis of your data.
- Build AI-enhanced business processes anywhere by using powerful API interfaces or reusable UI components.

Quick links

- Install: Install the service
- Set up: Set up the service after installation
- Upgrade: Upgrade the service
- Administer: Manage and maintain the service
- Develop apps: Write code and build applications

Installing Watson Discovery

An instance administrator can install Watson Discovery on IBM Cloud Pak® for Data Version 4.7.

Who needs to complete this task?

Instance administrator To install Watson Discovery, you must be an *instance administrator*. An instance administrator has permission to install software in the following projects:

The *operators project* for the instance

The operators for this instance of Cloud Pak for Data are installed in the operators project.

In the installation commands, the `${PROJECT_CPD_INST_OPERATORS}` environment variable refers to the operators project.

The *operands project* for the instance

The Cloud Pak for Data control plane and the services for this instance of Cloud Pak for Data are installed in the operands project.

In the installation commands, the `${PROJECT_CPD_INST_OPERANDS}` environment variable refers to the operands project.

When do you need to complete this task?

Review the following options to determine whether you need to complete this task:

- If you want to install the Cloud Pak for Data control plane and one or more services at the same time, follow the process in *Installing the platform and services* instead.
- If you didn't install Watson Discovery when you installed the Cloud Pak for Data control plane, complete this task to add Watson Discovery to your environment.
  Repeat as needed. If you are responsible for multiple instances of Cloud Pak for Data, you can repeat this task to install more instances of Watson Discovery on the cluster.

Information you need to complete this task

Review the following information before you install Watson Discovery:

Version requirements

All of the components that are associated with an instance of Cloud Pak for Data must be installed at the same release. For example, if the Cloud Pak for Data control plane is installed at Version 4.7.1, you must install Watson Discovery at Version 4.7.1.

Environment variables

The commands in this task use environment variables so that you can run the commands exactly as written.

- If you don't have the script that defines the environment variables, see *Setting up installation environment variables*.
- To use the environment variables from the script, you must source the environment variables before you run the commands in this task. For example, run:

```
source ./cpd_vars.sh
```

Security context constraint

Watson Discovery works with the default Red Hat® OpenShift® Container Platform security context constraint:

- On Version 4.10, the default SCC is `restricted`.
- On Version 4.12, the default SCC is `restricted-v2`

Storage requirements

You must specify storage classes when you install Watson Discovery. The following storage classes are recommended. However, if you don't use these storage classes on your cluster, ensure that you specify a storage class with an equivalent definition.

Storage	Notes	Storage classes
OpenShift Data Foundation	When you install the service, specify file storage and block storage.	• **File storage:** `ocs-storagecluster-cephfs` • **Block storage:** `ocs-storagecluster-ceph-rbd`
IBM® Storage Fusion Data Foundation	When you install the service, specify file storage and block storage.	• **File storage:** `ocs-storagecluster-cephfs` • **Block storage:** `ocs-storagecluster-ceph-rbd`
IBM Storage Fusion Global Data Platform	When you install the service, specify the same storage class for both file storage and block storage.	• **File storage:** `ibm-spectrum-scale-sc` • **Block storage:** `ibm-spectrum-scale-sc`
IBM Storage Scale Container Native	When you install the service, specify the same storage class for both file storage and block storage.	• **File storage:** `ibm-spectrum-scale-sc` • **Block storage:** `ibm-spectrum-scale-sc`
Portworx	When you install the service, the `--storage_vendor=portworx` option ensures that the service uses the correct storage classes.	• **File storage:** `portworx-rwx-gp3-sc` • **Block storage:** `portworx-db-gp2-sc`
NFS	Not supported.	Not applicable.

Storage	Notes	Storage classes
Amazon Elastic storage	When you install the service, specify file storage and block storage. File storage is provided by Amazon Elastic File System. Block storage is provided by Amazon Elastic Block Store.	**File storage:** `efs-nfs-client` **Block storage:** `gp2-csi` or `gp3-csi`
IBM Cloud storage	When you install the service, specify file storage and block storage. File storage is provided by IBM Cloud File Storage. Block storage is provided by IBM Cloud Block Storage.	**File storage:** `ibmc-file-gold-gid` or `ibm-file-custom-gold-gid` **Block storage:** `ibmc-block-gold`
NetApp Trident	When you install the service, specify the same storage class for both file storage and block storage.	**File storage:** `ontap-nas` **Block storage:** `ontap-nas`

Restriction: This service does not support the backup and restore methods provided by the underlying storage. For example, if you use IBM Storage Fusion for storage, the service cannot use IBM Spectrum® Protect Plus for backups.

Before you begin

This task assumes that the following prerequisites are met:

Prerequisite	Where to find more information
The cluster meets the minimum requirements for installing Watson Discovery.	See *System requirements*.
The workstation from which you will run the installation is set up as a client workstation and includes the following command-line interfaces: Cloud Pak for Data CLI: `cpd-cli` OpenShift CLI: `oc`	See *Setting up a client workstation*.
The Cloud Pak for Data control plane is installed.	See *Installing the platform and services*.
For environments that use a private container registry, such as air-gapped environments, the Watson Discovery software images are mirrored to the private container registry.	See *Mirroring images to a private container registry*.
For environments that use a private container registry, such as air-gapped environments, the `cpd-cli` is configured to pull the `olm-utils-v2` image from the private container registry.	See *Pulling the olm-utils-v2 image from the private container registry*.
The node settings are adjusted for Watson Discovery.	See *Changing required node settings*.
Multicloud Object Gateway is installed and configured.	See *Installing Multicloud Object Gateway*.
The secrets that enable Watson Discovery to connect to Multicloud Object Gateway exist.	See *Creating secrets for services that use Multicloud Object Gateway*.

Procedure

Complete the following tasks to install Watson Discovery:

1. Specifying configuration options
2. Installing the service
3. Validating the installation
4. What to do next

Specifying configuration options

You can specify the following installation option in the `install-options.yml` file in the `work` directory.

```
################################################################################
# Watson Discovery parameters
################################################################################
#discovery_deployment_type: Production
```

Property	Description

`discovery_deployment_type`	The deployment type for Watson™ Discovery.
	The deployment type determines the number of resources allocated to Watson Discovery.
	
	Default value
	`Production`
	Valid values
	`Production`
	A production deployment has at least two replicas of each pod to support production-scale workloads.
	`Starter`
	A starter deployment has fewer resources and less computing power than a production deployment.

Note: You cannot change the deployment type value later, but you can scale the sizes of the microservices individually later. For more information, see *Scaling Watson Discovery*.

Installing the service

To install Watson Discovery, complete the following steps:

1. Run the `cpd-cli manage login-to-ocp` command to log in to the cluster as a user with sufficient permissions to complete this task. For example:

   ```
   cpd-cli manage login-to-ocp \
   --username=${OCP_USERNAME} \
   --password=${OCP_PASSWORD} \
   --server=${OCP_URL}
   ```

   Tip: The `login-to-ocp` command takes the same input as the `oc login` command. Run `oc login --help` for details.
2. Run the following command to create the operator lifecycle manager (OLM) objects that are required by Red Hat OpenShift Container Platform for Watson Discovery to be added to the *operators project* for the instance:

   ```
   cpd-cli manage apply-olm \
   --release=${VERSION} \
   --cpd_operator_ns=${PROJECT_CPD_INST_OPERATORS} \
   --components=watson_discovery
   ```

   Wait for the `cpd-cli` to return the following message before you proceed to the next step:

   ```
   [SUCCESS]... The apply-olm command ran successfully
   ```

   If the `apply-olm` fails, see *Troubleshooting the apply-olm command during installation or upgrade*.
3. Create the custom resource for Watson Discovery.
   The command that you run depends on the storage on your cluster.

   - Red Hat OpenShift Data Foundation storage
     Run the appropriate command to create the custom resource.

     Default installation (without installation options)

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --license_acceptance=true
     ```

     Custom installation (with installation options)

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true
     ```

   - IBM Storage Fusion Data Foundation storage
     Run the appropriate command to create the custom resource.

     Default installation (without installation options)

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --license_acceptance=true
     ```

     Custom installation (with installation options)

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true
     ```

   - IBM Storage Fusion Global Data Platform storage
     Remember: When you use IBM Storage Fusion Global Data Platform storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class, typically `ibm-spectrum-scale-sc`.
     Run the appropriate command to create the custom resource.

     Default installation (without installation options)

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --license_acceptance=true
     ```

     Custom installation (with installation options)

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true
```

- IBM Storage Scale Container Native storage

  Remember: When you use IBM Storage Scale Container Native storage, both **`${STG_CLASS_BLOCK}`** and **`${STG_CLASS_FILE}`** point to the same storage class, typically **`ibm-spectrum-scale-sc`**.
  Run the appropriate command to create the custom resource.

  Default installation (without installation options)

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --license_acceptance=true
  ```

  Custom installation (with installation options)

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true
  ```

- Portworx storage

  Run the appropriate command to create the custom resource.

  Default installation (without installation options)

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --storage_vendor=portworx \
  --license_acceptance=true
  ```

  Custom installation (with installation options)

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --storage_vendor=portworx \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true
  ```

- AWS with EFS and EBS storage

  Run the appropriate command to create the custom resource.

  Default installation (without installation options)

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --license_acceptance=true
  ```

  Custom installation (with installation options)

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true
  ```

- IBM Cloud with IBM Cloud File Storage and IBM Cloud Block Storage

  Run the appropriate command to create the custom resource.

  Default installation (without installation options)

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  ```

```
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--license_acceptance=true
```

Custom installation (with installation options)

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true
```

- NetApp Trident

Remember: When you use NetApp Trident storage, both **`${STG_CLASS_BLOCK}`** and **`${STG_CLASS_FILE}`** point to the same storage class, typically **`ontap-nas`**.
Run the appropriate command to create the custom resource.

Default installation (without installation options)

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--license_acceptance=true
```

Custom installation (with installation options)

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true
```

Validating the installation

Watson Discovery is installed when the **`apply-cr`** command returns the following message:

```
[SUCCESS]... The apply-cr command ran successfully
```

If you want to confirm that the custom resource status is **`Completed`**, you can run the **`cpd-cli manage get-cr-status`** command. For example:

```
cpd-cli manage get-cr-status \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--components=watson_discovery
```

If you encounter any problems with the installation, check *Limitations and known issues in Watson Discovery* for a documented workaround.

What to do next

Watson Discovery is ready to be set up. See Post-installation setup for Watson Discovery for next steps.

Post-installation setup for Watson Discovery

After you install Watson Discovery, you must create an instance of the service.

You can create up to 10 service instances.

You can create a service instance in any of the following ways:

REST API
: To create an instance programmatically, see Creating a Watson Discovery service instance programmatically

Command line
: To create an instance by using the cpd-cli, see Creating a Watson Discovery service instance from the command line

Web client
: To create an instance from the web client user interface, see Creating a Watson Discovery service instance from the web client

What to do next

After you create a service instance, Watson Discovery is ready to use. For information about how to give people access to your instance, see Administering Watson Discovery.

Creating a Watson Discovery service instance programmatically

Use the application programming interface to create a Watson Discovery service instance.

This task is equivalent to provisioning a service instance from the Cloud Pak for Data web client. For more information, see Creating a Watson Discovery service instance from the web client. Alternatively, you can use the command-line interface to create a service instance. For more information, see Creating a Watson Discovery service instance from the command line.

Required role: To use this API, you must have the **can_provision** permission in Cloud Pak for Data.

Prerequisite step

Before you can use the API, you must get an API key that you can use to get an authorization token that you pass with the RESTful API request. Follow the steps that are described in *Generating API keys for authentication*.

Creating the service instance

To create a service instance by using the API, complete the following step.

1. The following command gets the hostname for the Cloud Pak for Data cluster where you want to create the instance.

   ```
   oc get route cpd -n $PROJECT_CPD_INSTANCE -ojsonpath='{.spec.host}'
   ```

   You will pass this hostname as the ***<host>*** variable in subsequent API requests.

2. Send a POST request to create a service instance.
   Update the following variables with the right values for your deployment:

   <host>
   : The hostname that you retrieved in the previous step.

   <token>
   : The authorization token that you obtained when you completed the prerequisite step.

   ${VERSION}
   : Environment variable for the version of the Cloud Pak for Data software that is installed.

   <instance-name>
   : Name that you want to use for the service instance. This name is displayed in the *Instances* page of the Cloud Pak for Data web client. The name can contain alphanumeric characters, spaces, dashes, underscores, and periods.

   ${PROJECT_CPD_INSTANCE}
   : Environment variable for the project name (formerly namespace) that was defined before the service was installed. For more information, see *Setting up installation environment variables*.

   ```
   curl --insecure --request POST --url <host>/zen-data/v3/service_instances \
     --header "Authorization: Bearer <token>" \
     --header 'Content-Type: application/json' \
     --data "{
       \"addon_type\": \"discovery\",
       \"addon_version\": \"${VERSION}\",
       \"create_arguments\": {
         \"deployment_id\": \"$PROJECT_CPD_INSTANCE-wd\",
         \"parameters\" : {
           \"serviceId\": \"discovery\",
           \"url\": \"https://wd-discovery-gateway.$PROJECT_CPD_INSTANCE.svc.cluster.local:60443/v2/service_instances\",
           \"watson\": true
          }
        },
        \"display_name\": \"<instance-name>\",
        \"namespace\": \"$PROJECT_CPD_INSTANCE\"
     }"
   ```

Example

The following example shows how to get the access token, and then make the API request to create the service instance.

1. Submit the following command to get the hostname for the Cloud Pak for Data cluster where you want to create the instance.

   ```
   oc get route cpd -n $PROJECT_CPD_INSTANCE -ojsonpath='{.spec.host}'
   ```

   The hostname that is returned in this example is `cpd.apps.my.cluster.example.com`. The hostname is then passed as the ***<host>*** variable in subsequent API requests.

2. Get a platform API key.
   a. Get an access token for the Cloud Pak for Data cluster. This request uses basic authentication; it provides the user credentials for a user in the administrative role who can provision service instances.

      ```
      curl --request POST --user jdoe:mypassword \
      --url <host>/v1/preauth/validateAuth
      ```

      The response contains an `accessToken` field that contains a bearer token. Store the token so that you can specify it in the next step.
   b. Generate an API key that you can use to identify yourself when you submit RESTful API requests. This request is authenticated with the token that was returned in the previous step.
      Note: The API key that is generated by this request replaces the existing API key that is associated with your user ID.

      ```
      curl --request POST \
      --url <host>/usermgmt/v1/user/apiKey \
      ```

```
    --header "Authorization: Bearer <token>"
```

This response contains an **apiKey** field that contains the API key. Store the key somewhere safe. You cannot recover this key if you lose it.

3. Use the API key to create a token that authorizes you to send API requests. Submit the following POST request to create the token.
   The request uses basic authentication; it provides the user credentials for the same user who created the API key and has permission to provision service instances.

```
curl --request POST --user jdoe:mypassword \
--url <host>/icp4d-api/v1/authorize
```

   If a message about a certificate error is displayed, you can add the `--insecure` argument to the request to disable certificate validation.
   In the body of the request, specify the API key that you obtained in the previous step. The user name must be the same name that is used to authenticate this request.

```
{
  "username":"jdoe",
  "api_key":"amgbHu4VwDXRho9yLvF2IN5QkUVMWykjAolxYYQp"
}
```

   The token that you will use to authenticate requests is returned in the `token` field of the response. Copy the token value so that you can use it later.

4. The following POST request creates the Watson Discovery service instance.
   In this example, the project name is `zen`.

   The URL that is specified in the `url` parameter in the request body is used by the Watson gateway microservice, which is a service that manages API requests for the Watson services.

   Example request:

```
curl --request POST --url https://cpd.apps.my.cluster.example.com/zen-data/v3/service_instances \
  --header "Authorization: Bearer <token>" \
  --header 'Content-Type: application/json' \
  --data "{
    \"addon_type\": \"discovery\",
    \"addon_version\": \"4.6.2\",
    \"create_arguments\": {
      \"deployment_id\": \"zen-wd\",
      \"parameters\" : {
        \"serviceId\": \"discovery\",
        \"url\": \"https://wd-discovery-gateway.zen.svc.cluster.local:60443/v2/service_instances\",
        \"watson\": true
       }
     },
     \"display_name\": \"My instance\",
     \"namespace\": \"zen\"
   }"
```

   The response contains an `id` parameter that contains the instance ID of the instance that is created. Make a note of the instance ID.

   Example response:

```
{
    "id": "1670881541354464"
}
```

Completing Watson Discovery tasks

After you create the instance, you can add projects and data by using the Watson Discovery APIs.

To use the APIs, complete the following steps:

1. To use the service APIs, submit requests to the appropriate endpoint for the method that you want to use by using the following syntax:

```
https://<url>/<method>
```

   You can authenticate the request by passing the same bearer token that you used to provision the service instance.

2. To construct the URL to use in the *`<url>`* segment of the request, replace the *`<host>`*, *`<deployment_id>`*, and *`<instance_id>`* segments of the URL with values from your deployment.

```
https://<host>/discovery/<deployment_id>/instances/<instance_id>/api
```

3. The *`method`* segment of the request defines the action that you want to take. To create a project, the method to use is `v2/projects`.
   To create a Document Retrieval project in the service instance that you created, you can send a POST request to the following endpoint:

```
https://<host>/discovery/<deployment_id>/instances/<instance_id>/api/v2/projects
```

   For example:

```
curl --request POST --url https://cpd.apps.my.cluster.example.com/discovery/zen-
wd/instances/1670881541354464/api/v2/projects?version=2020-08-30
```

   In the body of the request, define the project:

```
{
  "name": "My new project",
  "type": "document_retrieval"
}
```

For more information about the methods that you can use to work with Watson Discovery programmatically, see the Watson Discovery API documentation.

Listing the service instances

The following example syntax lists the Cloud Pak for Data service instances:

```
curl --request GET \
  --url https://<host>/zen-data/v3/service_instances \
  --header "Authorization: Bearer <token>"
```

Deleting the service instance

The following example syntax shows you how to delete a service instance that you specify by its instance ID:

```
curl --request DELETE \
  --url https://<host>/zen-data/v3/service_instances/<instanceId> \
  --header "Authorization: Bearer <token>"
```

Creating a Watson Discovery service instance from the command line

Use the command-line interface to create a Watson Discovery service instance.

This task is equivalent to provisioning a service instance from the Cloud Pak for Data web client. For more information, see Creating a Watson Discovery service instance from the web client. Alternatively, you can use the application programming interface to create a service instance. For more information, see Creating a Watson Discovery service instance programmatically.

Required role: To use the `create` command, you must have the **can_provision** permission in Cloud Pak for Data.

Prerequisite steps

Before you can use the Cloud Pak for Data command-line interface (cpd-cli) to create a service instance, you must complete the following steps:

1. Install the cpd-cli. For more information, see *Installing the Cloud Pak for Data command-line interface*.
2. Create a Cloud Pak for Data configuration profile name that has permission to provision instances. For more information, see *Creating a profile to use the management commands*.

Creating the service instance

To create a service instance by using the command-line interface, complete the steps described in the ***`service-instance.create`*** reference documentation.

The following procedure describes the steps for creating a Watson Discovery service instance in more detail.

1. Create a JSON file that defines the service instance that you want to create. Store the file in a location from which it can be referenced when you submit a command to the cpd-cli later.
   Use the following JSON snippet as a starting point, and then replace the values that are identified as variables.

```
{
        "addon_type": "discovery",
        "addon_version": "${VERSION}",
        "create_arguments": {
                "deployment_id": "${PROJECT_CPD_INSTANCE}-wd",
                "parameters": {
                        "serviceId": "discovery",
                        "url": "https://wd-discovery-
gateway.${PROJECT_CPD_INSTANCE}.svc.cluster.local:60443/v2/service_instances",
                        "watson": true


        "display_name": "<instance-name>",
        "namespace": "${PROJECT_CPD_INSTANCE}"
}
```

   Replace the following variables with values that reflect your deployment:

   `${VERSION}`
   Environment variable for the version of the Cloud Pak for Data software that is installed.
   `${PROJECT_CPD_INSTANCE}`
   Environment variable for the project name (formerly namespace) that was defined before the service was installed. For more information, see *Setting up installation environment variables*.
   `<instance-name>`
   Name that you want to use for the service instance. This name is displayed in the *Instances* page of the Cloud Pak for Data web client. The name can contain alphanumeric characters, spaces, dashes, underscores, and periods.

2. Use the following command to create the service instance:

```
cpd-cli service-instance create \
--profile <cpd-configuration-profile-name> --from-source <filename.json>
```

   Specify values for the following parameters:

   `<cpd-configuration-profile-name>`
   Specify the Cloud Pak for Data configuration profile that you created as a prerequisite step. For example, `cpd-admin-profile`.
   `<filename.json>`
   Specify the file path to the JSON file that you created in the previous step.

For example:

```
cpd-cli service-instance create \
--profile cpd-admin-profile --from-source ./wd-instance.json
```

Listing the service instances

The following example syntax lists the Cloud Pak for Data service instances that are associated with the specified profile:

```
cpd-cli service-instance \
--profile <cpd-configuration-profile-name> list
```

For example:

```
cpd-cli service-instance \
--profile cpd-admin-profile list
```

Getting a service instance

The following example syntax returns information about the Cloud Pak for Data service instance that you specify by name.

```
cpd-cli service-instance \
--profile <cpd-configuration-profile-name> get <instance-name>
```

For example:

```
cpd-cli service-instance \
--profile cpd-admin-profile get wd-test-instance
```

Deleting a service instance

The following example syntax shows you how to delete a service instance that you specify by name:

```
cpd-cli service-instance \
--profile <cpd-configuration-profile-name> delete <instance-name>
```

For example:

```
cpd-cli service-instance \
--profile cpd-admin-profile delete wd-test-instance
```

Creating a Watson Discovery service instance from the web client

Use the IBM Cloud Pak® for Data web client to create a Watson Discovery service instance.

Alternatively, you can use the command-line interface or the API to create a service instance. For more information, see Creating a Watson Discovery service instance from the command line or Creating a Watson Discovery service instance programmatically.

Permissions you need for this task:
You must be an administrator of the Red Hat® OpenShift® project.
Note: You must be the same person who installed the Watson Discovery service. You can provision an instance only for a Watson Discovery service that you installed.

To create an service instance, complete the following steps:

1. From the IBM Cloud Pak for Data web client menu, click **Services > Instances**.
2. From the instances page, click **New instance**.
3. Click the **Watson Discovery** service tile.
4. Click **New instance**.
   You can create a maximum of 10 instances. After you reach the maximum number, the New instance button is not displayed.
5. Add a name and other details for the service instance, and then click **Create**.
6. Wait 1 to 2 minutes for the new instance to be provisioned.

After the new instance is created, the web client shows information about the new service instance, including its URL and credentials in the *Access information* section of the *About this instance* page.

Upgrading Watson Discovery

The supported methods to upgrade Watson Discovery differ based on the Cloud Pak for Data version you're using.

- **Upgrading Watson Discovery from Version 4.5 to Version 4.7**
  An instance administrator can upgrade Watson Discovery from Cloud Pak for Data Version 4.5 to Version 4.7.
- **Upgrading Watson Discovery from Version 4.6 to Version 4.7**
  An instance administrator can upgrade Watson Discovery from Cloud Pak for Data Version 4.6 to Version 4.7.
- **Upgrading Watson Discovery from Version 4.7.x to a later 4.7 refresh**
  An instance administrator can upgrade Watson Discovery from Cloud Pak for Data Version 4.7.x to a later 4.7 refresh.

Upgrading Watson Discovery from Version 4.5 to Version 4.7

An instance administrator can upgrade Watson Discovery from Cloud Pak for Data Version 4.5 to Version 4.7.

Note: You cannot upgrade the Watson Discovery service by using the **`service-instance upgrade`** command from the Cloud Pak for Data command-line interface.

Who needs to complete this task?

Instance administrator To upgrade Watson Discovery, you must be an *instance administrator*. An instance administrator has permission to manage software in the following projects:

The *operators project* for the instance

The operators for this instance of Cloud Pak for Data are installed in the operators project. In the upgrade commands, the **`${PROJECT_CPD_INST_OPERATORS}`** environment variable refers to the operators project.

The *operands project* for the instance

The Cloud Pak for Data control plane and the services for this instance of Cloud Pak for Data are installed in the operands project. In the upgrade commands, the **`${PROJECT_CPD_INST_OPERANDS}`** environment variable refers to the operands project.

Review the following options to determine whether you need to complete this task:

If you want to upgrade the Cloud Pak for Data control plane and one or more services at the same time, follow the process in *Upgrading the platform and services* instead.
If you didn't upgrade Watson Discovery when you upgraded the Cloud Pak for Data control plane, complete this task to upgrade Watson Discovery.
Repeat as needed If you are responsible for multiple instances of Cloud Pak for Data, you can repeat this task to upgrade more instances of Watson Discovery on the cluster.

Information you need to complete this task

Review the following information before you upgrade Watson Discovery:

Version requirements

All the components that are associated with an instance of Cloud Pak for Data must be installed at the same release. For example, if the Cloud Pak for Data control plane is at Version 4.7.1, you must upgrade Watson Discovery to Version 4.7.1.

The commands in this task use environment variables so that you can run the commands exactly as written.

If you don't have the script that defines the environment variables, see *Setting up installation environment variables*.
To use the environment variables from the script, you must source the environment variables before you run the commands in this task. For example, run:

```
source ./cpd_vars.sh
```

Specify the storage that you use in your existing installation. You cannot change the storage that is associated with Watson Discovery during an upgrade. Ensure that the environment variables point to the correct storage classes for your environment.

Before you begin

This task assumes that the following prerequisites are met:

Prerequisite	Where to find more information
The cluster meets the minimum requirements for Watson Discovery.	See *System requirements*.
The workstation from which you will run the upgrade is set up as a client workstation and the following command-line interfaces:	See *Setting up a client workstation*.
The Cloud Pak for Data control plane is upgraded.	See *Upgrading the platform and services*.
For environments that use a private container registry, such as air-gapped environments, the Watson Discovery software images are mirrored to the private container registry.	See *Mirroring images to a private container registry*.
For environments that use a private container registry, such as air-gapped environments, the **`cpd-cli`** is configured to pull the **`olm-utils-v2`** image from the private container registry.	See *Pulling the olm-utils-v2 image from the private container registry*.
Multicloud Object Gateway is installed and configured.	See *Installing Multicloud Object Gateway*.
The secrets that enable Watson Discovery to connect to Multicloud Object Gateway exist.	See *Creating secrets for services that use Multicloud Object Gateway*.

Procedure

Complete the following tasks to upgrade Watson Discovery:

1. Back up your data.
   For more information, see *Backing up your data in Cloud Pak for Data* in the Watson Discovery product documentation.
2. Specifying configuration options
3. Upgrading the service
4. Validating that the upgrade was successful
5. Migrating to Multicloud Object Gateway from MinIO
6. Post-upgrade step

Note: Required only if you previously downgraded the RabbitMQ version to add FIPS compliance.

7. What to do next

Specifying installation options

You cannot change your deployment type on upgrade. Run the following command to determine the deployment type that is used in your existing Watson™ Discovery deployment:

```
oc get WatsonDiscovery wd \
--namespace=${PROJECT_CPD_INST_OPERANDS} \
-ojsonpath='{.spec.shared.deploymentType}'
```

- If the command returns `Production`, you don't need to specify any installation options. Proceed to *Upgrading the service*.
- If the command returns `Starter` or `Development`, complete the following steps to specify the deployment type installation option.
- If the command returns an empty response, the deployment is using the `Development` deployment type. Complete the following steps to specify the deployment type installation option.

To set the deployment type installation option:

1. Create a file called `install-options.yml` in the `cpd-cli` `work` directory.
2. Add the appropriate entry to the file:
   - For the `Starter` deployment type, add:

     ```
     discovery_deployment_type: Starter
     ```

   - For the `Development` deployment type, add:

     ```
     discovery_deployment_type: Development
     ```

Upgrading the service

Important: The Operator Lifecycle Manager (OLM) objects for Watson Discovery were updated when you upgraded the Cloud Pak for Data platform. The `cpd-cli manage apply-olm` updates all of the OLM objects in the *operators project* at the same time.
To upgrade Watson Discovery:

1. Run the `cpd-cli manage login-to-ocp` command to log in to the cluster as a user with sufficient permissions to complete this task. For example:

   ```
   cpd-cli manage login-to-ocp \
   --username=${OCP_USERNAME} \
   --password=${OCP_PASSWORD} \
   --server=${OCP_URL}
   ```

   Tip: The `login-to-ocp` command takes the same input as the `oc login` command. Run `oc login --help` for details.
2. Update the custom resource for Watson Discovery.
   The command that you run depends on the storage on your cluster:

   - Red Hat OpenShift Data Foundation storage
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - IBM Storage Fusion Data Foundation storage
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - IBM Storage Fusion Global Data Platform storage
     Remember: When you use IBM Storage Fusion storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class, typically `ibm-spectrum-scale-sc`.
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     ```

```
--file_storage_class=${STG_CLASS_FILE} \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true \
--upgrade=true
```

- IBM Storage Scale Container Native storage

  Remember: When you use IBM Storage Scale Container Native storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class, typically `ibm-spectrum-scale-sc`.
  Run the following command to create the custom resource.

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true \
  --upgrade=true
  ```

- Portworx storage

  Run the following command to create the custom resource.

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --storage_vendor=portworx \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true \
  --upgrade=true
  ```

- AWS with EFS and EBS storage

  Run the following command to create the custom resource.

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true \
  --upgrade=true
  ```

- IBM Cloud with IBM Cloud File Storage and IBM Cloud Block Storage

  Run the following command to create the custom resource.

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true \
  --upgrade=true
  ```

Validating the upgrade

Watson Discovery is upgraded when the `apply-cr` command returns the following message:

```
[SUCCESS]... The apply-cr command ran successfully
```

If you want to confirm that the custom resource status is `Completed`, you can run the `cpd-cli manage get-cr-status` command:

```
cpd-cli manage get-cr-status \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--components=watson_discovery
```

Migrating to Multicloud Object Gateway from MinIO

Starting in IBM Cloud Pak® for Data Version 4.7, Multicloud Object Gateway replaces the MinIO data store. After you upgrade Watson Discovery, you must enable the service to use Multicloud Object Gateway and remove the MinIO data store.

Important: The service will be quiesced while the data migration occurs, which means that the service will be unavailable for the duration of the migration.

1. Apply the following patch to enable Multicloud Object Gateway:

   ```
   oc patch WatsonDiscover wd \
   --namespace=${PROJECT_CPD_INST_OPERANDS} \
   type=merge \
   --patch='{"spec":{"shared":{"mcg":{"enabled": true}}}}'
   ```

2. Confirm that the status of the Watson Discovery service is shown as **READY**.

   ```
   oc get watsonDiscovery wd \
   --namespace=${PROJECT_CPD_INST_OPERANDS}
   ```

3. Enter the following command to remove MinIO.

   ```
   oc patch WatsonDiscovery wd \
   --namespace=${PROJECT_CPD_INST_OPERANDS} \
   --type merge \
   --patch '{"spec": {"minio": {"create": false}}}'
   ```

   Wait a few minutes for the MinIO cluster to be deleted.

4. Run the following command to ensure that the MinIO cluster was removed. It should not return any resource.

   ```
   oc get miniocluster wd-minio \
   --namespace=${PROJECT_CPD_INST_OPERANDS}
   ```

   Repeat this step until no resources are returned.

5. Run the following commands to delete remaining resources of MinIO.

   ```
   oc delete pvc \
   --namespace=${PROJECT_CPD_INST_OPERANDS} \
   -l app.kubernetes.io/instance=wd-minio
   oc delete pvc wd-minio-discovery-backup-pvc \
   --namespace=${PROJECT_CPD_INST_OPERANDS}
   oc delete secret wd-discvery-kes-client-secret \
   --namespace=${PROJECT_CPD_INST_OPERANDS}
   ```

Upgrading RabbitMQ on FIPS clusters

Attention: Complete this step only if you changed the version of RabbitMQ to enable FIPS support in the 4.5.x release.
Previously, the RabbitMQ message broker was not FIPS-compliant. The workaround was to downgrade the version of the operator to 3.8. RabbitMQ 3.9 is complaint. Therefore, if you previously took steps to downgrade the version, take an extra step now to reverse the change.

1. Run the following command to verify the patch has been applied:

   ```
   --namsespace flag.
   oc get WatsonDiscovery wd \
   --namespace=${PROJECT_CPD_INST_OPERANDS} \
   -o jsonpath='{.spec.rabbitmq.version}'
   ```

If the command returns 3.8 then you will need to remove the hard-coded version setting. A user with administrative privileges must complete the following step:

1. Run the following command to remove the field from the custom resource so that the appropriate version value can be specified by the operator:

   ```
   oc patch WatsonDiscovery wd \
   --namespace=${PROJECT_CPD_INST_OPERANDS} \
   --type=json \
   --patch='[{"op": "remove", "path": "/spec/rabbitmq/version"}]'
   ```

What to do next

Watson Discovery is ready to use. For information about how to give people access to your instance, see Administering Watson Discovery.

Upgrading Watson Discovery from Version 4.6 to Version 4.7

An instance administrator can upgrade Watson Discovery from Cloud Pak for Data Version 4.6 to Version 4.7.

Note: You cannot upgrade the Watson Discovery service by using the `service-instance upgrade` command from the Cloud Pak for Data command-line interface.

Who needs to complete this task?

Instance administrator To upgrade Watson Discovery, you must be an *instance administrator*. An instance administrator has permission to manage software in the following projects:

The *operators project* for the instance

The operators for this instance of Cloud Pak for Data are installed in the operators project. In the upgrade commands, the `${PROJECT_CPD_INST_OPERATORS}` environment variable refers to the operators project.

The *operands project* for the instance

The Cloud Pak for Data control plane and the services for this instance of Cloud Pak for Data are installed in the operands project. In the upgrade commands, the `${PROJECT_CPD_INST_OPERANDS}` environment variable refers to the operands project.

When do you need to complete this task?

Review the following options to determine whether you need to complete this task:

- If you want to upgrade the Cloud Pak for Data control plane and one or more services at the same time, follow the process in *Upgrading the platform and services* instead.
- If you didn't upgrade Watson Discovery when you upgraded the Cloud Pak for Data control plane, complete this task to upgrade Watson Discovery.
  Repeat as needed If you are responsible for multiple instances of Cloud Pak for Data, you can repeat this task to upgrade more instances of Watson Discovery on the cluster.

Information you need to complete this task

Review the following information before you upgrade Watson Discovery:

Version requirements
: All the components that are associated with an instance of Cloud Pak for Data must be installed at the same release. For example, if the Cloud Pak for Data control plane is at Version 4.7.1, you must upgrade Watson Discovery to Version 4.7.1.

Environment variables
: The commands in this task use environment variables so that you can run the commands exactly as written.

  If you don't have the script that defines the environment variables, see *Setting up installation environment variables*.
  To use the environment variables from the script, you must source the environment variables before you run the commands in this task. For example, run:

```
source ./cpd_vars.sh
```

Storage requirements
: Specify the storage that you use in your existing installation. You cannot change the storage that is associated with Watson Discovery during an upgrade. Ensure that the environment variables point to the correct storage classes for your environment.

Before you begin

This task assumes that the following prerequisites are met:

Prerequisite	Where to find more information
The cluster meets the minimum requirements for Watson Discovery.	If this task is not complete, see *System requirements.z*
The workstation from which you will run the upgrade is set up as a client workstation and the following command-line interfaces:	If this task is not complete, see Updating clientz workstations.
The Cloud Pak for Data control plane is upgraded.	If this task is not complete, see *Upgrading the platformz and services.*
For environments that use a private container registry, such as air-gapped environments, the Watson Discovery software images are mirrored to the private container registry.	If this task is not complete, see *Mirroring images to az private container registry.*
For environments that use a private container registry, such as air-gapped environments, the `cpd-cli` is configured to pull the `olm-utils-v2` image from the private container registry.	If this task is not complete, see *Pulling the olm-utils-v2z image from the private container registry.*
Multicloud Object Gateway is installed and configured.	If this task is not complete, see *Installing Multicloudz Object Gateway.*
The secrets that enable Watson Discovery to connect to Multicloud Object Gateway exist.	If this task is not complete, see *Creating secrets forz services that use Multicloud Object Gateway.*

Procedure

Complete the following tasks to upgrade Watson Discovery:

1. Back up your data.
   For more information, see *Backing up your data in Cloud Pak for Data* in the Watson Discovery product documentation.
2. Specifying configuration options
3. Upgrading the service
4. Validating that the upgrade was successful
5. Removing MinIO
6. What to do next

Specifying installation options

You cannot change your deployment type on upgrade. Run the following command to determine the deployment type that is used in your existing Watson™ Discovery deployment:

```
oc get WatsonDiscovery wd \
--namespace=${PROJECT_CPD_INST_OPERANDS} \
-ojsonpath='{.spec.shared.deploymentType}'
```

- If the command returns `Production`, you don't need to specify any installation options. Proceed to *Upgrading the service*.
- If the command returns `Starter` or `Development`, complete the following steps to specify the deployment type installation option.
- If the command returns an empty response, the deployment is using the `Development` deployment type. Complete the following steps to specify the deployment type installation option.

To set the deployment type installation option:

1. Create a file called `install-options.yml` in the `cpd-cli` `work` directory.
2. Add the appropriate entry to the file:
   - For the `Starter` deployment type, add:

     ```
     discovery_deployment_type: Starter
     ```

   - For the `Development` deployment type, add:

     ```
     discovery_deployment_type: Development
     ```

Upgrading the service

Important: The Operator Lifecycle Manager (OLM) objects for Watson Discovery were updated when you upgraded the Cloud Pak for Data platform. The `cpd-cli manage apply-olm` updates all of the OLM objects in the *operators project* at the same time.
To upgrade Watson Discovery:

1. Run the `cpd-cli manage login-to-ocp` command to log in to the cluster as a user with sufficient permissions to complete this task. For example:

   ```
   cpd-cli manage login-to-ocp \
   --username=${OCP_USERNAME} \
   --password=${OCP_PASSWORD} \
   --server=${OCP_URL}
   ```

   Tip: The `login-to-ocp` command takes the same input as the `oc login` command. Run `oc login --help` for details.
2. Update the custom resource for Watson Discovery.
   The command that you run depends on the storage on your cluster:

   - Red Hat OpenShift Data Foundation storage
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - IBM Storage Fusion Data Foundation storage
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - IBM Storage Fusion Global Data Platform storage
     Remember: When you use IBM Storage Fusion storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class, typically `ibm-spectrum-scale-sc`.
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - IBM Storage Scale Container Native storage
     Remember: When you use IBM Storage Scale Container Native storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class, typically `ibm-spectrum-scale-sc`.
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - Portworx storage
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --storage_vendor=portworx \
     --param-file=/tmp/work/install-options.yml \
     ```

```
--license_acceptance=true \
--upgrade=true
```

- AWS with EFS and EBS storage

  Run the following command to create the custom resource.

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true \
  --upgrade=true
  ```

- IBM Cloud with IBM Cloud File Storage and IBM Cloud Block Storage

  Run the following command to create the custom resource.

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true \
  --upgrade=true
  ```

- NetApp Trident

  Remember: When you use NetApp Trident storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class.
  Run the following command to create the custom resource.

  ```
  cpd-cli manage apply-cr \
  --components=watson_discovery \
  --release=${VERSION} \
  --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
  --block_storage_class=${STG_CLASS_BLOCK} \
  --file_storage_class=${STG_CLASS_FILE} \
  --param-file=/tmp/work/install-options.yml \
  --license_acceptance=true \
  --upgrade=true
  ```

Validating the upgrade

Watson Discovery is upgraded when the `apply-cr` command returns:

```
[SUCCESS]... The apply-cr command ran successfully
```

If you want to confirm that the custom resource status is `Completed`, you can run the `cpd-cli manage get-cr-status` command:

```
cpd-cli manage get-cr-status \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--components=watson_discovery
```

Removing MinIO

Starting in IBM Cloud Pak® for Data Version 4.7, Multicloud Object Gateway replaces the MinIO data store. After you upgrade Watson Discovery, you must remove the MinIO data store.

1. Confirm that the status of the Watson Discovery service is shown as `READY`.

   ```
   oc get watsonDiscovery wd \
   --namespace=${PROJECT_CPD_INST_OPERANDS}
   ```

2. Enter the following command to remove MinIO.

   ```
   oc patch WatsonDiscovery wd \
   --namespace=${PROJECT_CPD_INST_OPERANDS} \
   --type merge \
   --patch '{"spec": {"minio": {"create": false}}}'
   ```

   Wait a few minutes for the MinIO cluster to be deleted.

3. Run the following command to ensure that the MinIO cluster was removed. It should not return any resource.

   ```
   oc get miniocluster wd-minio \
   --namespace=${PROJECT_CPD_INST_OPERANDS}
   ```

   Repeat this step until no resources are returned.

4. Run the following commands to delete remaining resources of MinIO.

   ```
   oc delete pvc \
   --namespace=${PROJECT_CPD_INST_OPERANDS} \
   -l app.kubernetes.io/instance=wd-minio
   ```

```
oc delete pvc wd-minio-discovery-backup-pvc \
--namespace=${PROJECT_CPD_INST_OPERANDS}
oc delete secret wd-discvery-kes-client-secret \
--namespace=${PROJECT_CPD_INST_OPERANDS}
```

What to do next

Watson Discovery is ready to use. For information about how to give people access to your instance, see Administering Watson Discovery.

Upgrading Watson Discovery from Version 4.7.x to a later 4.7 refresh

An instance administrator can upgrade Watson Discovery from Cloud Pak for Data Version 4.7.x to a later 4.7 refresh.

Who needs to complete this task?
: Instance administrator To upgrade Watson Discovery, you must be an *instance administrator*. An instance administrator has permission to manage software in the following projects:

The *operators project* for the instance
: The operators for this instance of Cloud Pak for Data are installed in the operators project. In the upgrade commands, the **`${PROJECT_CPD_INST_OPERATORS}`** environment variable refers to the operators project.

The *operands project* for the instance
: The Cloud Pak for Data control plane and the services for this instance of Cloud Pak for Data are installed in the operands project. In the upgrade commands, the **`${PROJECT_CPD_INST_OPERANDS}`** environment variable refers to the operands project.

When do you need to complete this task?
: Review the following options to determine whether you need to complete this task:

- If you want to upgrade the Cloud Pak for Data control plane and one or more services at the same time, follow the process in *Upgrading an instance of Cloud Pak for Data* instead.
- If you didn't upgrade Watson Discovery when you upgraded the Cloud Pak for Data control plane, complete this task to upgrade Watson Discovery.
  Repeat as needed If you are responsible for multiple instances of Cloud Pak for Data, you can repeat this task to upgrade more instances of Watson Discovery on the cluster.

Information you need to complete this task

Review the following information before you upgrade Watson Discovery:

Version requirements
: All the components that are associated with an instance of Cloud Pak for Data must be installed at the same release. For example, if the Cloud Pak for Data control plane is at Version 4.7.1, you must upgrade Watson Discovery to Version 4.7.1.

Environment variables
: The commands in this task use environment variables so that you can run the commands exactly as written.

- If you don't have the script that defines the environment variables, see *Setting up installation environment variables.*
- To use the environment variables from the script, you must source the environment variables before you run the commands in this task. For example, run:

  ```
  source ./cpd_vars.sh
  ```

Storage requirements
: Specify the storage that you use in your existing installation. You cannot change the storage that is associated with Watson Discovery during an upgrade. Ensure that the environment variables point to the correct storage classes for your environment.

Before you begin

This task assumes that the following prerequisites are met:

Prerequisite	Where to find more information
The cluster meets the minimum requirements for Watson Discovery.	See *System requirements.*
The workstation from which you will run the upgrade is set up as a client workstation and the following command-line interfaces: • Cloud Pak for Data CLI: **`cpd-cli`** • OpenShift® CLI: **`oc`**	See *Updating clientz workstations.*
The Cloud Pak for Data control plane is upgraded.	See *Upgrading an instance ofzz Cloud Pak for Data.*
For environments that use a private container registry, such as air-gapped environments, the Watson Discovery software images are mirrored to the private container registry.	See *Mirroring images to azz private container registry.*
For environments that use a private container registry, such as air-gapped environments, the **`cpd-cli`** is configured to pull the **`olm-utils-v2`** image from the private container registry.	See *Pulling the olm-utils-v2zz image from the private container registry.*

Procedure

Complete the following tasks to upgrade Watson Discovery:

1. Specifying configuration options

Specifying configuration options

When you upgrade the service, the same deployment type must be used as was used for the previous deployment. A `production` deployment type is configured for 4.7 deployments by default.

If the operator notices a change in the deployment type during the upgrade, it will automatically correct it. Alternatively, you can configure the upgrade to create a `starter` deployment type by specifying installation options.

If the operator corrects the deployment type, you'll see this message in the `status` field of the affected instance:

```
Deployment type is changed: development -> production.
Mutating ''spec.shared.deploymentType'' is not allowed.
Setting deploymentType back to last known deployment type: development'
reason: ConfigError
status: "False"
```

Upgrading the service

Important: The Operator Lifecycle Manager (OLM) objects for Watson Discovery were updated when you upgraded the Cloud Pak for Data platform. The `cpd-cli manage apply-olm` updates all of the OLM objects in the *operators project* at the same time.
To upgrade Watson Discovery:

1. Run the `cpd-cli manage login-to-ocp` command to log in to the cluster as a user with sufficient permissions to complete this task. For example:

   ```
   cpd-cli manage login-to-ocp \
   --username=${OCP_USERNAME} \
   --password=${OCP_PASSWORD} \
   --server=${OCP_URL}
   ```

   Tip: The `login-to-ocp` command takes the same input as the `oc login` command. Run `oc login --help` for details.
2. Update the custom resource for Watson Discovery.
   The command that you run depends on the storage on your cluster:

   - Red Hat OpenShift Data Foundation storage
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - IBM Storage Fusion Data Foundation storage
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - IBM Storage Fusion Global Data Platform storage
     Remember: When you use IBM Storage Fusion storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class, typically `ibm-spectrum-scale-sc`.
     Run the following command to create the custom resource.

     ```
     cpd-cli manage apply-cr \
     --components=watson_discovery \
     --release=${VERSION} \
     --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
     --block_storage_class=${STG_CLASS_BLOCK} \
     --file_storage_class=${STG_CLASS_FILE} \
     --param-file=/tmp/work/install-options.yml \
     --license_acceptance=true \
     --upgrade=true
     ```

   - IBM Storage Scale Container Native storage
     Remember: When you use IBM Storage Scale Container Native storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class, typically `ibm-spectrum-scale-sc`.

Run the following command to create the custom resource.

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true \
--upgrade=true
```

- Portworx storage

  Run the following command to create the custom resource.

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--storage_vendor=portworx \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true \
--upgrade=true
```

- AWS with EFS and EBS storage

  Run the following command to create the custom resource.

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true \
--upgrade=true
```

- IBM Cloud with IBM Cloud File Storage and IBM Cloud Block Storage

  Run the following command to create the custom resource.

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true \
--upgrade=true
```

- NetApp Trident

  Remember: When you use NetApp Trident storage, both `${STG_CLASS_BLOCK}` and `${STG_CLASS_FILE}` point to the same storage class.z
  Run the following command to create the custom resource.

```
cpd-cli manage apply-cr \
--components=watson_discovery \
--release=${VERSION} \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--block_storage_class=${STG_CLASS_BLOCK} \
--file_storage_class=${STG_CLASS_FILE} \
--param-file=/tmp/work/install-options.yml \
--license_acceptance=true \
--upgrade=true
```

Validating the upgrade

Watson Discovery is upgraded when the `apply-cr` command returns:

```
[SUCCESS]... The apply-cr command ran successfully
```

If you want to confirm that the custom resource status is `Completed`, you can run the `cpd-cli manage get-cr-status` command:

```
cpd-cli manage get-cr-status \
--cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS} \
--components=watson_discovery
```

What to do next

Watson Discovery is ready to use. For information about how to give people access to your instance, see Administering Watson Discovery.

Administering Watson Discovery

Learn about the ways that an administrator can keep Watson Discovery running smoothly.

An administrator can perform the following tasks:

- Giving users access to a Watson Discovery instance
- Managing security for your Watson Discovery data stores
- Monitoring the PostgreSQL datastore for Watson Discovery
- Scaling Watson Discovery

For more information about how to back up and restore a Watson Discovery deployment, see one of the following topics:

Perform an offline backup of the Watson™ Discovery deployment by using scripts provided by the service: *Backing up and restoring data*
Perform an online backup of the Cloud Pak for Data and Watson Discovery using OpenShift® APIs for Data Protection: *Backing up and restoring data*

Giving users access to a Watson Discovery instance

After you install and provision a Watson Discovery instance, you can share the URL for the product user interface with other people. However, those users can log in to the product user interface only if you give them access.

Before you can give someone access to a Watson Discovery instance, the person must have access to use IBM Cloud Pak® for Data itself. For more information, see *Managing access to the platform*.

The following table describes the service roles.

Role	Actions available
Admin	• Start and use the product user interface. • Submit API requests. • Give users access to deployed instances. • Delete instances.
User	• Start and use the product user interface. • Submit API requests.

To give people access to an instance of the Watson Discovery service, a person with the Admin role must complete the following steps:

1. From the Cloud Pak for Data web client menu, click Services > Instances.
2. Find your service instance, and then click the overflow menu, and choose Manage access.
3. Click Add users.
4. Search for the people that you want to add, and then select them.
   Note: If people that you want to add are not listed, make sure they have permission to access Cloud Pak for Data, as explained earlier.
5. Assign a role to each person.
6. Click Add users.

Managing security for your Watson Discovery data stores

You can manage the access credentials for your MinIO, PostgreSQL, Elasticsearch, RabbitMQ, and etcd data stores by creating secrets objects for each data store. Secrets are generated automatically during installation. You can create new credentials after installation. You can also rotate your credentials at any time for added security. Creating secrets objects for your data stores is optional.

Permissions you need for these tasks:
    You must be an administrator of the Red Hat® OpenShift® project to manage the cluster.

Updating secrets objects for your data stores

The following procedures describe how to update the secrets objects for each data source individually. Use these steps to change the secrets for your data stores after you install the service or to rotate the secrets for added security.
Note: Because Watson Discovery depends on third-party data stores, they are limited by the password policies that the data stores enforce. For more information about restrictions on the length of passwords and the characters they can include, see the documentation for the data stores.

Prerequisite step

Before you re-create secrets objects for MinIO, PostgreSQL, Elasticsearch, and RabbitMQ, you must shut down the Watson Discovery service.

Attention: You cannot re-create or rotate the credentials for the etcd operator unless the etcd service is running and operational. Do not complete the procedure to re-create the secrets object for etcd until after theWatson Discovery service is restarted.

1. Back up the data stores by using the backup script that is described in *Backup and restore data*. Store the backups in a safe location.
2. Run the following command to ensure that you're logged in to the correct namespace, the installation is complete, and the service is stable:

   ```
   oc get WatsonDiscovery wd -o jsonpath='{.status.watsonDiscoveryStatus}'
   ```

   The service is stable when the command returns the status `Completed`.
3. Shut down the Watson Discovery by editing your custom resource with the following command:

   ```
   oc patch WatsonDiscovery wd --type=merge \
   --patch='{"spec": {"shutdown": "true"}}'
   ```

Save the change to the custom resource.

4. Wait for the Watson Discovery to shut down. To check the status of the services, enter the following commands:

   ```
   oc get WatsonDiscovery wd -o jsonpath='{.status.customResourceQuiesce}'
   ```

   ```
   oc get WatsonDiscovery wd -o jsonpath='{.status.datastoreQuiesce}'
   ```

   The services are ready when the commands return the status `QUIESCED`.

5. Use the procedures in the following sections to create new secrets objects for the data stores.

   Note: Change the names of only the passwords and secrets. Do not change the usernames that are associated with those passwords and secrets.

Creating a secrets object for your PostgreSQL data store

1. Complete the prerequisite step to quiesce the service first.
2. Create the new secret by using the `oc create secret` command.

   ```
   oc create secret generic new-auth-secret-name \
   --from-literal=password=new-postgres-password
   ```

   where the following values are specified:
   - *`new-auth-secret-name`* is a new secret name, such as `credentials-psql`.
   - *`new-postgres-password`* is replaced by the new password value.
3. To confirm that the new value was saved successfully, you can use the following command:

   ```
   oc extract secret/new-auth-secret-name --to=-
   ```

4. Create a patch to apply the new secret to the service.

   ```
   oc patch WatsonDiscovery wd --type=merge \
   --patch='{"spec":{"postgres":{"authSecretName": "auth-secret-name"}}}'
   ```

5. Now that new credentials exist, you can delete the previous secret.

   ```
   oc delete secret old-auth-secret-name
   ```

   To delete the secrets that are generated by the installation process, use the following command:

   ```
   oc delete secret wd-discovery-cn-postgres-su wd-discovery-cn-postgres-wd
   ```

Creating a secrets object for your RabbitMQ data store

1. Complete the prerequisite step to quiesce the service first.
2. Create the new secret by using the `oc create secret` command.

   ```
   oc create secret generic new-auth-secret-name \
   --from-literal=rabbitmq-password=new-rabbitmq-password \
   --from-literal=rabbitmq-management-password=new-rabbitmq-mgmt-password
   ```

   where the following values are specified:
   - *`new-auth-secret-name`* is a new secret name, such as `credentials-rmq`.
   - *`new-rabbitmq-password`* is replaced by the new password value.
   - *`new-rabbitmq-mgmt-password`* is replaced by the new management password value.
3. To confirm that the new values were saved successfully, you can use the following command:

   ```
   oc extract secret/new-auth-secret-name --to=-
   ```

4. Create a patch to apply the new secret to the service.

   ```
   oc patch WatsonDiscovery wd --type=merge \
   --patch='{"spec":{"rabbitmq":{"authSecretName": "new-auth-secret-name"}}}'
   ```

5. Now that new credentials exist, you can delete the previous secret.

   ```
   oc delete secret old-auth-secret-name
   ```

   To delete the secret that is generated by the installation process, use the following command:

   ```
   oc delete secret wd-discovery-rabbitmq-auth
   ```

Creating a secrets object for your MinIO data store

1. Complete the prerequisite step to quiesce the service first.
2. Create the new secret by using the `oc create secret` command.

   ```
   oc create secret generic new-auth-secret-name \
   --from-literal=accesskey=new-access-key --from-literal=secretkey=new-secret-key \
   --from-literal=sseMasterKey=wire-master-key:new-master-key
   ```

   where the following values are specified:
   - *`new-auth-secret-name`* is a new secret name, such as `credentials-minio`.
   - *`new-access-key`* is replaced by the new access key value.
   - *`new-secret-key`* is replaced by the new secret key value.
   - *`new-master-key`* is replaced by the new master key value.
3. To confirm that the new values were saved successfully, you can use the following command:

```
oc extract secret/new-auth-secret-name --to=-
```

4. Create a patch to apply the new secret to the service.

   ```
   oc patch WatsonDiscovery wd --type=merge \
   --patch='{"spec":{"minio":{"authSecretName": "new-auth-secret-name"}}}'
   ```

5. Now that new credentials exist, you can delete the previous secret.

   ```
   oc delete secret old-auth-secret-name
   ```

   To delete the secret that is generated by the installation process, use the following command:

   ```
   oc delete secret wd-discovery-minio-auth
   ```

Creating a secrets object for your Elasticsearch data store

1. Complete the prerequisite step to quiesce the service first.
2. Create the new secret by using the `oc create secret` command.

   ```
   oc create secret generic new-auth-secret-name \
   --from-literal=password=new-elastic-password
   ```

   where the following values are specified:
   - *`new-auth-secret-name`* is a new secret name, such as `credentials-elastic`.
   - *`new-elastic-password`* is replaced by the new password value.
3. To confirm that the new value was saved successfully, you can use the following command:

   ```
   oc extract secret/new-auth-secret-name --to=-
   ```

4. Create a patch to apply the new secret to the service.

   ```
   oc patch WatsonDiscovery wd --type=merge \
   --patch='{"spec":{"elasticsearch":{"authSecretName": "new-auth-secret-name"}}}'
   ```

5. Now that new credentials exist, you can delete the previous secret.

   ```
   oc delete secret old-auth-secret-name
   ```

   To delete the secrets that are generated by the installation process, use the following command:

   ```
   oc delete secret wd-discovery-elastic-secret wd-discovery-elastic-secret-es
   ```

Restarting the service

1. Restart the service by applying a patch that changes the `shutdown` status for the service to false.

   ```
   oc patch WatsonDiscovery wd --type=merge --patch='{"spec": {"shutdown": "false"}}'
   ```

2. Wait for the services to restart, and then run the following commands to check the status:

   ```
   oc get WatsonDiscovery wd -o jsonpath='{.status.customResourceQuiesce}'
   ```

   ```
   oc get WatsonDiscovery wd -o jsonpath='{.status.datastoreQuiesce}'
   ```

   The service is fully restarted when the `NOT_QUIESCED` status is displayed.

Creating a secrets object for your etcd data store

Do not create a secrets object or perform any rotation of credentials for the etcd service while the service is quiesced or shut down. Only when the service is up and running, complete the following steps to re-create the secrets object for etcd.

The etcd operator requires both credentials to exist in the namespace to perform rotation, do not delete the existing credentials until after the new credentials are created.

1. Verify that the etcd pods are up and running.

   ```
   oc get pods -lapp=etcd,app.kubernetes.io/instance=wd
   ```

2. Create the new secret by using the `oc create secret` command.

   ```
   oc create secret generic new-auth-secret-name \
   --from-literal=password=new-etcd-password
   ```

   where the following values are specified:
   - *`new-auth-secret-name`* is a new secret name, such as `credentials-etcd`.
   - *`new-etcd-password`* is replaced by the new password value.
3. To confirm that the new value was saved successfully, you can use the following command:

   ```
   oc extract secret/new-auth-secret-name --to=-
   ```

4. Create a patch to apply the new secret to the service.

   ```
   oc patch WatsonDiscovery wd --type=merge \
   --patch='{"spec":{"etcd":{"authSecretName": "new-auth-secret-name"}}}'
   ```

5. Monitor the etcd pods as they restart after the credential is rotated internally.

   ```
   oc get pods -lapp=etcd,app.kubernetes.io/instance=wd
   ```

This change causes other Watson Discovery pods to restart as they adopt the new credential.

6. Monitor the other service pods as they restart.

```
oc get WatsonDiscovery wd -w
```

Monitoring the PostgreSQL datastore for Watson Discovery

You can enable monitoring of the PostgreSQL datastore to receive updates on its usage and status by the Watson Discovery. The events can be consumed by Prometheus monitoring software or whatever application you use for monitoring.

Permissions you need for these tasks:
You must have login credentials for the cluster.
You must have the name of the project (the namespace) for your deployment, which you can obtain from your cluster administrator.

Monitoring the PostgreSQL datastore

By enabling monitoring for user-defined projects in addition to the default platform monitoring, you can monitor your own projects with the Red Hat® OpenShift® Container Platform monitoring stack.

1. Create a configuration map that defines a list of the queries that you want to monitor.

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    cnpg.io/reload: "true"
  name: {name-of-config-map}
data:
  {name-of-query}: |
  ...
```

where *{name-of-config-map}* is the name of the configuration map and *{name-of-query}* is the name of the query followed by a list of query examples.
For example, the following configuration map is named `postgresql-wds-record-monitor` and checks for queries that run for over 5 minutes.

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    cnpg.io/reload: "true"
  name: postgresql-wds-record-monitor
data:
  custom-queries: |
    wd_long_running_query:
      primary: true
      query: "select datname,count(*) as count from pg_stat_activity where (now() - pg_stat_activity.query_start)>
interval '5 minutes' group by datname"
      metrics:
        - datname:
            usage: "LABEL"
            description: "Name of the database"
        - count:
            usage: "GAUGE"
            description: "Number of queries that have been running for over 5 minutes"
```

2. Patch the Discovery service custom resource to enable EDB monitoring by using the configuration map that you defined in the previous step.

```
oc patch wd wd --type=merge --patch '{"spec": {"postgres": {"monitoring": {\
"customQueriesConfigMap": [{"key": "{name-of-query}", "name": "{name-of-config-map}"}]}}}}'
```

This patch causes the Discovery operator to add the new configuration map to `spec.monitoring.customQueriesConfigMap` in the PostgreSQL custom resource.

3. Create the `cluster-monitoring-config` configuration map and set `enableUserWorkload` to `true` under `data/config.yaml`. (If the configmap already exists, this command overwrites it.) When set to `true`, the `enableUserWorkload` parameter enables monitoring for user-defined projects in a cluster.

```
cat << EOF | oc apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    enableUserWorkload: true
EOF
```

4. Set `openshift.io/user-monitoring` to `true` for the IBM Cloud Pak® for Data project (namespace). This enables user monitoring when the Network Policy is created.

```
oc label namespace ${PROJECT_CPD_INST_OPERANDS} openshift.io/user-monitoring=true
```

5. Create a Network Policy to allow Prometheus to monitor the `PodMonitor` resource in the instance namespace. This allows the project to accept connections from the Red Hat OpenShift Container Platform monitoring stack.

```
cat << EOF | oc apply -f -
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: wd-discovery-cn-postgres-prometheus
  namespace: ${PROJECT_CPD_INST_OPERANDS}
spec:

  - from:
    - namespaceSelector:
       matchLabels:
         network.openshift.io/policy-group: monitoring
  podSelector:
    matchLabels:
      postgresql: wd-discovery-cn-postgres
  policyTypes:
  - Ingress
```

6. Create a `PodMonitor` configuration map to create a `PodMonitor` object that watches the PostgreSQL pods for metrics and makes them available to the Red Hat OpenShift Container Platform monitoring stack.

   ```
   cat << EOF | oc apply -f -
   apiVersion: monitoring.coreos.com/v1
   kind: PodMonitor
   metadata:
     name: wd-discovery-cn-postgres-podmonitor
   spec:
     podMetricsEndpoints:
     - port: metrics
     selector:
       matchLabels:
         postgresql: wd-discovery-cn-postgres
   ```

7. Ensure that you can see the PostgreSQL metrics from your monitoring stack. For a list of predefined metrics that are exposed by PostgreSQL and for information about how to define your own metrics, see Monitoring in the *EDB PostgreSQL for Kubernetes* documentation.

Scaling Watson Discovery

After you install the service, you can change settings such as the number of replicas of the microservices to use, or the replica size to increase or decrease the capacity ofz the application.

In a multitenant environment, you install the Watson Discovery service one time, and then you can deploy up to 10 separate instances of the service. The computingz resources, such as CPU and memory, that are provisioned for the installation are shared by all of the deployed instances. For planning purposes, consider the total size ofz artifacts, such as collections and enrichments, from across all of the service instances, not the size per instance.

Unless you specify otherwise, when you install Watson Discovery, a production-ready deployment is created. It has at least two replicas of each pod to supportz production-scale workloads. However, if you have complicated projects or high volumes of use, you might want to increase the capacity of your deployment. Alternatively,z if you have low usage, you might want to decrease the capacity of your deployment to reduce costs.

You can use the `scaleConfig` setting in the service custom resource (CR) to set the scaling configuration for a service. For more information, see Manually scalingz resources for services.

Alternatively, you can change the configuration of a deployment by changing the YAML file that is associated with the deployment.

To edit the YAML file, you must have administrative privileges for the project where the deployment is hosted. To change the file, you use one of the following commands:

Edit command
: Opens an editor where you can make changes to the YAML definition. The command has the following syntax:

  ```
  oc edit WatsonDiscovery wd
  ```

Patch command
: Patches an existing resource by merging the configuration details from a local YAML file into the existing configuration. The command has the following syntax:

  ```
  oc patch WatsonDiscovery wd --patch "$(cat $path-to-file)" --type='merge'
  ```

Scaling the number of replicas

To change the number of replicas that are used by a microservice, edit the custom resource. The operator applies the change to the current deployment for you.

1. Create a YAML file that specifies the number of replicas for the target pods.
   For example, to increase the number of API Gateway pods to `3` , create a file that is named `patch.yaml` with the following content:

   ```
   spec:
     api:
       replicas: 3
   ```

2. Apply the patch to the custom resource by using the patch command. For example:

   ```
   oc patch WatsonDiscovery wd --type=merge --patch "$(cat patch.yaml)"
   ```

Sample YAML file

The following YAML file sample shows values that you can change.

```
spec:
  tooling:
    minerapp: # Content mining application
```

```
      replicas: 2
    tooling: # Discovery product user interface
      replicas: 2
  api:  # API gateway pod
    replicas: 2
  hdp:  # Hadoop worker
    worker:
      replicas: 3
  ingestion:  # Crawler pod
    crawler:
      replicas: 2
  statelessapi:  # Analyze API
    runtime:
      replicas: 2
  wksml:  # Machine learning model enrichment
    replicas: 2
```

Scaling an existing persistent volume claim size

A persistent volume claim (PVC) is a request for a piece of storage (a persistent volume) in a cluster. To change the resources that are used by a Discovery microservice, you must edit the claim. After you change the details of your request, the cluster manages how to make the increased or reduced amount of resources available to the service.

To scale your deployment of Watson Discovery, complete the following steps:

1. From the system where the service is running, edit the YAML file for the resource that you want to change.
   a. Use the following command to list the persistent volume claims that are in use by Watson Discovery:

   ```
   oc get pvc --selector 'app.kubernetes.io/name in (wd, discovery)'
   ```

   b. Look for the PVC name for the resource that you want to change, and then use the following command to open its associated YAML file for editing:

   ```
   oc edit pvc {PVC NAME FROM PREVIOUS STEP}
   ```

   To edit the configuration of the Elastic search resource, the command might look like this, for example:

   ```
   oc edit pvc data-wd-ibm-elasticsearch-es-server-client-0
   ```

   c. Edit the YAML file for the PVC pod.
   The following settings only can be changed for a persistent volume claim (PVC) after the PVC is created.

   ```
     elasticsearch:
       clientNode:
         persistence:
           size: 1Gi     # PVC size for Elasticsearch client node. Cannot be changed in custom resource.
       dataNode:
         persistence:
           size: 40Gi     # PVC size for Elasticsearch data node. Cannot be changed in custom resource.
       masterNode:
         persistence:
           size: 2Gi     # PVC size for Elasticsearch coordinator node. Cannot be changed in custom resource.
     etcd:
       storageSize: 10Gi     # PVC size for etcd data store. Cannot be changed in custom resource.
     minio:
       persistence:
         size: 100Gi     # PVC size for MinIO. Cannot be changed in custom resource.
     postgres:
       database:
         storageRequest: 30Gi     # PVC size for the PostgreSQL. Cannot be changed in custom resource.
     rabbitmq:
       persistentVolume:
         size: 5Gi     # PVC size for the RabbitMQ data store. Cannot be changed in custom resource.
   ```

   In the following example, the storage size for the Elastic search client node is changed from 1 Gi to 2 Gi:

   ```
   spec:
     ...
     elasticsearch:
       clientNode:
         persistence:
           size: 2Gi
   ```

   d. After you edit the YAML file for the PVC, save and close it.
   Give the cluster a few minutes to pick up and apply the changes.
2. To verify that the changes were applied, run the following command:

   ```
   oc get pvc {PVC NAME}
   ```

Important: You cannot make this change persist. For example, you cannot update the custom resource for the service with these changes to apply them automatically. Instead, you must repeat the previous steps each time a persistent volume claim is added as a result of increasing the number of replicas, for example.

Preparing for system maintenance

If you want to scale down the service entirely so that you can perform system maintenance, for example, you can do so.

1. Back up the system data so that you can restore the service if you run into any problems. For more information, see Backing up and restoring data.
2. Stop the service.

   ```
   oc patch WatsonDiscovery wd --type=merge \
   --patch='{"spec":{"shutdown": "true"}}'
   ```

All of the pods that are associated with the service are quiesced, except for the PostgreSQL pods. The PostgreSQL service does not support a quiesce operation. Instead, these pods are set into a maintenance mode called *fencing* where data is protected while the system is maintained. In fencing mode, the pods continue to run in a non-ready state.

3. To enable Red Hat® OpenShift® Container Platform to drain the worker nodes fully, the PostgreSQL pods must be stopped. Use the following command to stop and drain the nodes:

   ```
   oc adm drain
   ```

4. Complete your maintenance tasks.
5. After system maintenance is finished, restart the service.

   ```
   oc patch WatsonDiscovery wd --type=merge \
   --patch='{"spec":{"shutdown": "false"}}'
   ```

Uninstalling Watson Discovery

An instance administrator can uninstall Watson Discovery.

Who needs to complete this task?
Instance administrator To uninstall Watson Discovery, you must be an *instance administrator*. An instance administrator has permission to manage software in the following projects:

The *operators project* for the instance
The operators for this instance of Cloud Pak for Data are installed in the operators project. In the uninstall commands, the `${PROJECT_CPD_INST_OPERATORS}` environment variable refers to the operators project.

The *operands project* for the instance
The Cloud Pak for Data control plane and the services for this instance of Cloud Pak for Data are installed in the operands project. In the uninstall commands, the `${PROJECT_CPD_INST_OPERANDS}` environment variable refers to the operands project.

When do you need to complete this task?
Complete this task if you want to remove Watson Discovery from an instance of Cloud Pak for Data.

Repeat as needed If you are responsible for multiple instances of Cloud Pak for Data, you can repeat this task to remove other instances of Watson Discovery on the cluster.

Information you need to complete this task

Review the following information before you uninstall Watson Discovery:

Environment variables
The commands in this task use environment variables so that you can run the commands exactly as written.

- If you don't have the script that defines the environment variables, see *Setting up installation environment variables*.
- To use the environment variables from the script, you must source the environment variables before you run the commands in this task. For example, run:

  ```
  source ./cpd_vars.sh
  ```

Procedure

Complete the following tasks to uninstall Watson Discovery:

1. Uninstalling the service
2. Removing associated resources

Uninstalling the service

To uninstall Watson Discovery:

1. Run the `cpd-cli manage login-to-ocp` command to log in to the cluster as a user with sufficient permissions to complete this task. For example:

   ```
   cpd-cli manage login-to-ocp \
   --username=${OCP_USERNAME} \
   --password=${OCP_PASSWORD} \
   --server=${OCP_URL}
   ```

   Tip: The `login-to-ocp` command takes the same input as the `oc login` command. Run `oc login --help` for details.
2. Create an environment variable named `OPERAND_NAME` to store the operand name for Watson Discovery.

   ```
   oc get WatsonDiscovery --namespace ${PROJECT_CPD_INST_OPERANDS}
   export OPERAND_NAME=<operand_name>
   ```

3. Delete the `wd` custom resource for Watson Discovery.

   ```
   cpd-cli manage delete-cr \
   --components=watson_discovery \
   --cpd_instance_ns=${PROJECT_CPD_INST_OPERANDS}
   ```

   Wait for the `cpd-cli` to return the following message before you proceed to the next step:

   ```
   [SUCCESS]... The delete-cr command ran successfully
   ```

4. Delete the OLM objects for Watson Discovery:

   ```
   cpd-cli manage delete-olm-artifacts \
   --cpd_operator_ns=${PROJECT_CPD_INST_OPERATORS} \
   --components=watson_discovery
   ```

   Wait for the `cpd-cli` to return the following message:

   ```
   [SUCCESS]... The delete-olm-artifacts command ran successfully
   ```

Removing associated resources

Complete these extra steps to clean up after the service is removed:

1. Run the following command to delete a resource that is not labeled.

   ```
   oc delete pvc wd-minio-discovery-backup-pvc
   ```

2. Run the following command to list the resources that were created by Watson Discovery and verify that they are deleted. The `-l` parameter is a lowercase L that is used to request a list.

   ```
   oc get 
   all,wdall,role,rolebinding,secret,serviceaccount,rabbitmqcluster,miniocluster,etcdcluster,cluster.postgresql.k8s.enterpris
   edb.io,\
   watsongateway,modeltrain,certificate,elasticsearchcluster -l 'icpdsupport/addOnId=discovery'
   ```

   If Watson Discovery was uninstalled successfully, the response is empty. If it is not, delete any remaining resources by using the `oc delete` ***`resource-type resource-name`*** command.

3. Verify that all persistent volume claim (PVC) resources that were created by Watson Discovery are deleted. Run the following commands to list any remaining persistent volume claim (PVC) resources:

   ```
   oc get pvc -l 'app.kubernetes.io/name in (wd,discovery)'
   oc get pvc -l 'etcd_cluster=wd-discovery-etcd'
   ```

   If any remaining PVC resources are listed, delete them by running the following command for each resource in turn:

   ```
   oc delete pvc pvc-name
   ```

4. Remove Watson Discovery data from Multicloud Object Gateway by completing the following steps:
   a. Download the MinIO client.

      Refer to the MinIO product documentation and follow the appropriate instructions for your platform.

   b. Get the `host` and `port` information for the Multicloud Object Gateway and store it in an environment variable named `MCG_ENDPOINT` by entering the following command:

      ```
      export MCG_ENDPOINT="$(oc get route s3 --namespace openshift-storage \
      --output jsonpath='https://{.spec.host}')"
      ```

   c. Get the access key and secret key of Multicloud Object Gateway from the `noobaa-account-watson-discovery` secret and store the keys in environment variables.

      ```
      export MCG_ACCESS_KEY="$(oc extract secret/noobaa-account-watson-discovery \
      --keys=AWS_ACCESS_KEY_ID --to=- --namespace ${PROJECT_CPD_INST_OPERANDS})"
      ```

      ```
      export MCG_SECRET_KEY="$(oc extract secret/noobaa-account-watson-discovery \
       --keys=AWS_SECRET_ACCESS_KEY --to=- \
       --namespace ${PROJECT_CPD_INST_OPERANDS})"
      ```

   d. Configure the MinIO client for Multicloud Object Gateway.

      ```
      mc alias set discovery "${MCG_ENDPOINT}" "${MCG_ACCESS_KEY}" "${MCG_SECRET_KEY}"
      ```

      Note: If necessary, follow the instructions in the MinIO documentation to configure a security certificate. Otherwise, specify the `--insecure` option to disable certificate verification. If you disable verification for the client, you must include the `--insecure` option when you submit the `mc` commands.
   e. Run the following command to confirm that the MinIO client is successfully configured. The command lists the buckets that are used by Watson Discovery.

      ```
      mc ls discovery
      ```

      If an error is displayed, check the variables that you defined in the previous steps and ensure that the Multicloud Object Gateway host and port are accessible from your console.
   f. Delete the following objects and buckets that were created by Watson Discovery:
      - `common`
      - `cnm`
      - `elastic-backup`
      - `exported-documents`
      - `ranker-wire-all`
      - `sdu`
      - `sentence-classification-input`
      - `sentence-classification-results`

      To do so, do one of the following things:
      - Run the following command one time for each item in the list and replace `<name>` with a name from the list:

        ```
        mc rm --recursive --force discovery/<name>-${PROJECT_CPD_INST_OPERANDS}-${OPERAND_NAME}
        mc rb discovery/<name>-${PROJECT_CPD_INST_OPERANDS}-${OPERAND_NAME}
        ```

      - First, define the buckets.

```
MCG_BUCKETS=( "common" "sdu" "sentence-classification-input" \
"sentence-classification-results" "exported-documents" "elastic-backup" "ranker-wire-all" "cnm" )
```

Next, remove all of the buckets with one command.

```
for bucket in "${MCG_BUCKETS[@]}" ; do \
mc rm --recursive --force discovery/${bucket}-${PROJECT_CPD_INST_OPERANDS}-${OPERAND_NAME} ; \
mc rb discovery/${bucket}-${PROJECT_CPD_INST_OPERANDS}-${OPERAND_NAME} ; \
done
```

g. Remove the Multicloud Object Gateway configuration from the MinIO client by entering the following command:

```
mc alias remove discovery
```

5. Verify that all secret resources that were created by the service are deleted. Run the following commands to list any remaining resources:

```
oc get secret wd-discovery-cert-manager-tls wd-discovery-kes-client-secret noobaa-account-watson-discovery noobaa-cert-
watson-discovery
```

6. If any resources are listed, delete them by running the following command for each resource one at a time:

```
oc delete secret {secret-name}
```