

*Installation guide for CP4BA 25.0.0  
Workflow Process Service Authoring and  
Runtime*



---

# Contents

<b>Preparing your chosen capabilities.....</b>	<b>1</b>
Preparing databases and secrets for your chosen capabilities by running a script.....	1
Preparing to install Workflow Process Service Authoring.....	12
<b>Creating a production deployment.....</b>	<b>13</b>
Generating the custom resource with the deployment script.....	14
Checking and completing your custom resource.....	16
Checking the cluster configuration.....	16
Configuring Workflow Process Service Authoring.....	19
Validating the YAML in your custom resource file.....	20
Deploying the custom resource you created with the deployment script.....	21
Optional: Installing network policies.....	24
Validating your CP4BA production deployment objects.....	27
<b>Completing post-installation tasks.....</b>	<b>30</b>
Workflow Process Service Authoring.....	30
Cloud Pak for Business Automation foundation.....	32
Business Automation Studio.....	32
Cloud Pak foundational services.....	32
ROKS on IBM Cloud.....	34
<b>Optional: Installing IBM Workflow Process Service Runtime production     deployments.....</b>	<b>35</b>
<b>Optional: Configuring Workflow Process Service Runtime to work with Workflow     Process Service Authoring.....</b>	<b>46</b>

## Preparing your chosen capabilities

---

Each capability that you want to install must be prepared before you apply your custom resource.

### About this task

**Important:** If you want to use EDB Postgres as your database, then you must use the `cp4a-prerequisites.sh` script for your CP4BA installation to specify the request for the operator to create the database.

The following progress bar shows you where you are in the installation process.



Preparing a capability can involve many steps and more often than not involves setting up persistence and taking measures to secure the containers.

### Important:

Take note of the capabilities and the optional components that you plan to install. Many capabilities include components from other patterns, so you must search for and use the preparing tasks under these capability headings to make sure that your installation is successful. For more information, see [Capability patterns for production deployments](#).

Remember, Business Automation Navigator (BAN) is always installed in every deployment, so you need to create the database (ICNDB) and its secret (ibm-ban-secret). As a minimum, you need a supported database before you can apply a custom resource.

## Preparing databases and secrets for your chosen capabilities by running a script

---

The `cp4a-prerequisites.sh` script is provided in the `cert-kubernetes` repository to help you prepare for an installation of Cloud Pak for Business Automation. The script generates property files for the selected capabilities in your deployment and must be run before your deployment is installed.

### Before you begin

Before you use the `cp4a-prerequisites.sh` script to generate the property files, make sure that you review the requirements for the capabilities that you want to install together with your target database. This information is normally found in the preparing sections for each capability, where you can find the steps to manually create the databases. Consider your intended workload and the number of users that you want to access the services. For operational and performance reasons, it is important that network latency between the applications and the database server is as small as possible. For deployments that need to operate continuously with no interruptions in service, enable the databases for high availability (HA).

For more information about the supported databases, see the [Software Product Compatibility Reports](#).

**Tip:** Run the `cp4a-prerequisites.sh` script in the "property" mode to create the property files for your selected capabilities and database. Then, take a note of the properties in these files so that you can match up these values with the configuration of your database services.

The `cp4a-prerequisites.sh` script uses the following utility tools and needs them to be installed on your client machine.

**Note:** If the script detects any of the required tools are missing on the client, it reports the names and versions of the tools. It then provides a choice for you to install them.

- kubectl (the version that matches your Red Hat® OpenShift® cluster version)

If you prepared your client machine for an online deployment, then you kubectl is already installed.

- Java™ runtime environment (JRE 8.x is installed by the script if it is not found)

- Java

- keytool – Make sure that you add the keytool to your system PATH.

- [OpenSSL](#)

Create an environment variable to locate the target CP4BA namespace (cp4ba-project). Before you run the command, you must be logged in to your Red Hat OpenShift cluster.

```
export NAMESPACE=<cp4ba-project>
```

**Note:** If you deployed the operators in a separate namespace to the target CP4BA namespace, you must create the secrets in the CP4BA deployment namespace. For example, if you have cp4a-operators and cp4a-operands namespaces, then use the latter. Otherwise, use the name of the target CP4BA namespace.

## About this task

Instead of going through the many documented steps to create the databases and secrets for the capabilities in your Cloud Pak for Business Automation deployment, you can use a script to generate the SQL statement files (scripts) and YAML template files for the secrets.

The cp4a-prerequisites.sh script has three modes.

### property

The property mode supports the generation of property files for multiple database servers. The script uses a "**DB\_SERVER\_LIST**" key in the cp4ba\_db\_server.property file to list the number of instances, and creates the user property files (cp4ba\_user\_profile.property, cp4ba\_db\_name\_user.property, cp4ba\_db\_server.property, and cp4ba\_LDAP.property). Review and modify these files to match your infrastructure. Add values for the database server name, database names, database schema, LDAP server name, and LDAP attributes.

### generate

The generate mode uses the modified property files to generate the DB SQL statement file and the YAML template for the secret.

### validate

The validate mode checks whether the generated databases and the secrets are correct and ready to use in a CP4BA deployment.

After you downloaded cert-kubernetes, change the directory to the scripts folder under cert-kubernetes/scripts. For more information about downloading cert-kubernetes, see [Preparing a client to connect to the cluster](#).

The script can be run from this location and has the following options:

```
Usage: cp4a-prerequisites.sh -m [modetype] -n [CP4BA_NAMESPACE]
Options:
  -h Display help
  -m The valid mode types are: [property], [generate], or [validate]
  -n The target namespace of the CP4BA deployment.
      STEP1: Run the script in [property] mode. Creates property files (DB/LDAP property file)
with default values (database name/user).
      STEP2: Modify the DB/LDAP/user property files with your values.
      STEP3: Run the script in [generate] mode. Generates the DB SQL statement files and YAML
templates for the secrets based on the values in the property files.
      STEP4: Create the databases and secrets by using the modified DB SQL statement files and
YAML templates for the secrets.
```

STEP5: Run the script in [validate] mode. Checks whether the databases and the secrets are created before you install CP4BA.

All three modes can be run on the same client machine, but you can also run the property and generate modes on different clients. If you want to use different clients, then copy the temporary property file from the property mode with the output folder to the other client. Make a copy of the following files and put them into the downloaded cert-kubernetes folder on the other client:

```
cert-kubernetes/scripts/.tmp/.TEMPORARY.property  
cert-kubernetes/cp4ba-prerequisites/project/$NAMESPACE
```

**Note:** Some properties use an absolute path in their values. If you do copy the script to a different machine, make sure that the absolute paths are updated to match the location of the copied script.

The values of the following properties need to be modified after you copy the cp4ba-prerequisites folder to a different client.

```
*****cp4ba_db_server.property*****  
<DB_PREFIX_NAME>.DATABASE_SSL_CERT_FILE_FOLDER  
  
*****cp4ba_LDAP_server.property*****  
LDAP_SSL_CERT_FILE_FOLDER  
  
*****cp4ba_user_profile.property*****  
APP_ENGINE.SESSION_REDIS_SSL_CERT_FILE_FOLDER
```

If you ran the cp4a-prerequisites.sh -m generate command on the original client, you must run the command again after you modified the property files to re-create the SSL secret templates with the updated absolute paths.

## Procedure

1. Make sure that you downloaded the cert-kubernetes repository to a Linux® based machine (CentOS Stream/RHEL/MacOS) or a client to a Linux-based machine.
2. Make sure that you are in the scripts folder under cert-kubernetes.
3. Log in to the target cluster.

Using the Red Hat OpenShift CLI:

```
oc login https://<cluster-ip>:<port> -u <cluster-user> -p <password>
```

On ROKS, if you are not already logged in:

```
oc login --token=<token> --server=https://<cluster-ip>:<port>
```

4. Run the script in the "property" mode.

```
./cp4a-prerequisites.sh -m property -n $NAMESPACE
```

The \$NAMESPACE is the target project for the CP4BA deployment.

Follow the prompts in the command window to enter the required information.

- a. Select the Cloud Pak for Business Automation capabilities that you want to install.
- b. Select the optional components that you want to include.
- c. Select Yes if you want to enable FIPS for your Cloud Pak for Business Automation deployment.

**Tip:** The script asks this question only if you asked to check that FIPS is enabled on the cluster in the cluster admin script. The response is stored in the cp4ba-fips-status configMap.

If you select Yes, the script creates the **CP4BA.ENABLE\_FIPS** property in the cp4ba\_user\_profile.property file and records your selection.

```
CP4BA.ENABLE_FIPS="true"
```

If you select No, the value is stored as false.

The property determines the value of the `shared_configuration.enable_fips` parameter in the custom resource.

- d. Choose the LDAP type that you want to use for the CP4BA deployment.

By default, the LDAP is SSL enabled. You can disable SSL for the LDAP when you edit the LDAP property file. The script shows the following message:

```
[*] You can change the property "LDAP_SSL_ENABLED" in the property file
"cp4ba_LDAP.property" later. "LDAP_SSL_ENABLED" is "TRUE" by default.
```

- e. Enter your dynamic storage classes for slow, medium, fast file storage (RWX).  
f. Enter a block storage class name (RWO).  
g. Select a deployment profile size from small, medium, or large [1 to 3]. The default is small (1).  
h. Choose the database type that you want to use for the CP4BA deployment.

**Note:** If you select EDB Postgres, the CP4BA operator creates and initializes the database instances.

If you select a different database type and it is not an external PostgreSQL, the CP4BA operator creates and initializes EDB Postgres databases for Document Processing and Automation Decision Services. You do not need to complete anything in the property files for these instances.

The script sets the following fields in the `cp4ba_db_server.property` file.

- `postgresql-edb.DATABASE_SERVERNAME="postgres-cp4ba-rw.{{ meta.namespace }}.svc"`
- `postgresql-edb.DATABASE_PORT="5432"`
- `postgresql-edb.DATABASE_SSL_SECRET_NAME="{{ meta.name }}-pg-client-cert-secret"`

The CP4BA operator creates a cluster custom resource (CR) so that the EDB Postgres operator can create the EDB Postgres instance. The CP4BA operator generates the secret `"{{ meta.name }}-pg-client-cert-secret"` based on the root CA, which is set in the **`shared_configuration.root_ca_secret`** parameter.

The script also generates the `cp4ba_db_name_user.property` file, which is used to define the database server name, username, and password. The script uses this property file to create the data source section in the CP4BA CR and generate the secret templates. If the database names are not specified in the CR, the operator uses the default database names for each data source that uses the EDB Postgres instance. The same applies to the username and password for each database. If the username and password exists in the secret for each component, then the operator creates the user in the EDB Postgres instance with the password that is specified in the secret. If the username or password, or both, do not exist in the secret for a component, then a default username and password are used for that database. If the operator uses the default username and password, it also updates the secrets for each component.

After the EDB Postgres operator is created, the CP4BA operator does not manage the EDB Postgres instance and does not change it in any way. Your system administrator can manage the EDB cluster instance by following the [EDB Postgres documentation](#). For more information about backing up the EDB Postgres instance, see [Backing up EDB Postgres](#).

By default, the databases are SSL enabled. You can disable SSL for a database when you edit the database property file. The script shows the following message:

```
[*] You can change the property "DATABASE_SSL_ENABLE" in the property file
"cp4ba_db_server.property" later. "DATABASE_SSL_ENABLE" is "TRUE" by default.
```

- i. **If required:** If you selected a database type other than EDB Postgres, then enter the alias names for all the database servers to be used by the CP4BA deployment. For example,

dbserver1,dbserver2 sets two database servers. The first server is named dbserver1 and the second server is named dbserver2.

**Note:** The database server names cannot contain a dot (.) character.

- j. If you chose an external PostgreSQL database type for your deployment, you can also choose to use the database for Platform UI (Zen) and Identity Management (IM) as the metastore DB. Collect the relevant database information so you can enter the required values in the generated property files. For more information, see [Setting up an external PostgreSQL database server for IM](#) and [Configuring an external PostgreSQL database for Zen](#).

**Remember:** Zen stores users, groups, service instances, vault integration, and secret references in the metastore DB.

**Note:** If you select **Yes** to choose an external PostgreSQL DB, the script generates properties with the prefix CP4BA.ZEN\_EXTERNAL\_POSTGRES\_DATABASE and CP4BA.IM\_EXTERNAL\_POSTGRES\_DATABASE in the cp4ba\_user\_profile.property file.

If you chose any other type of database, then the CP4BA operator creates and initializes the embedded cloud-native EDB Postgres database instances.

- k. Choose whether to use an external TLS certificate for the OpenSearch and Kafka deployments. The OpenSearch and Kafka operators can be configured to use an external TLS certificate instead of the default root CA. Copy the certificates into the dedicated folder for your database (cert-kubernetes/scripts/cp4ba-prerequisites/project/\$NAMESPACE/propertyfile/cert/db/<database folder>).

If you select **No**, the CP4BA operator creates the leaf certificates based on the default root CA. The CP4BA operator checks the namespace for the cp4ba-tls-issuer resource, and if it exists, it then creates the necessary certificates for Kafka and OpenSearch.

When the script is finished, the messages include some actions. Read the next actions carefully and make sure that you complete them all before you go to the next step.

```
===== Created all property files for CP4BA =====
[NEXT ACTIONS]
Enter the <Required> values in the property files under <extracted_cert-kubernetes_archive>/
cert-kubernetes/scripts/cp4ba-prerequisites/project/<CP4BA_NAMESPACE>/propertyfile
[*] The key name in the property file is created by the cp4a-prerequisites.sh and is NOT
EDITABLE.
[*] The value in the property file must be within double quotes.
[*] The value for User/Password in [cp4ba_db_name_user.property]
[cp4ba_user_profile.property] file should NOT include special characters: single quotation
""
[*] The value in [cp4ba_LDAP.property] or [cp4ba_External_LDAP.property]
[cp4ba_user_profile.property] file should NOT include special character '''
```

The propertyfile directory has the following file structure:

```
├─ cert
│   └─ db
│       ├── <db_server_alias1>
│       └─ ...
└─ ldap
├─ cp4ba_LDAP.property
├─ cp4ba_db_name_user.property
├─ cp4ba_db_server.property
└─ cp4ba_user_profile.property
```

**Note:** The db directory contains one or more db server alias names that you specified.

If you plan to enable SSL-based connections for your database or LDAP servers, you must copy the SSL certificate from your remote server to the propertyfile folder structure.

- The SSL certificate for the LDAP server must be named ldap-cert.crt and copied to the folder cp4ba-prerequisites/project/<cp4ba\_namespace>/propertyfile/cert/ldap.
- If you plan to enable SSL-based connections for your PostgreSQL database server, use the following guidance.

- If you use both server and client authentication, retrieve the server certificate, client certificate, and client private key from your database server. Copy them into the folder `cp4ba-prerequisites/project/<cp4ba_namespace>/propertyfile/cert/db/<DB_ALIAS_NAME>`. The files must be named `root.crt`, `client.crt`, and `client.key`.
  - If you use server-only authentication, retrieve the server certificate from your database server and copy it into the folder `cp4ba-prerequisites/project/<cp4ba_namespace>/propertyfile/cert/db/<DB_ALIAS_NAME>`. The certificate must be named `db-cert.crt`.
  - If you plan to enable SSL-based connections for any other database server type, retrieve the server certificate file from your remote database server and copy it into the folder `cert-kubernetes/scripts/cp4ba-prerequisites/project/prod/propertyfile/cert/db/<DB_ALIAS_NAME>`. The certificate must be named `db-cert.crt`.
5. Make sure that you are in the `propertyfile` folder under `cp4ba-prerequisites/project/$NAMESPACE` and edit the property files as indicated by the NEXT ACTIONS messages from the script. Update the (`cp4ba_db_name_user.property`, `cp4ba_db_server.property`, `cp4ba_LDAP.property`, `cp4ba_user_profile.property`, and optionally `cp4ba_External_LDAP.property`) with the values in your environment.

**Important:** Use the `{Base64}` prefix for LTPA, KEYSTORE, and LDAP\_BIND\_DN passwords that have special characters. The following example shows that the `{Base64}` prefix must be used and the password must be encoded when it has a special character.

```
LDAP_BIND_DN_PASSWORD="{Base64}UGFzc3dvcmQk" # Which is "Password$" when the Base64 string
is decoded.
LTPA_PASSWORD="{Base64}UGFzc3dvcmQk" # Which is "Password$" when the Base64 string is
decoded.
KEYSTORE_PASSWORD="{Base64}UGFzc3dvcmQk" # Which is "Password$" when the Base64 string is
decoded.
```

- a. Edit the global section in the `cp4ba_user_profile.property` file, and then the other sections for each capability that you selected.

The global section contains license properties and the needed storage classes. The FIPS enablement property and the egress property to generate network policies are also present. The BAN section is always included, and the users and groups must be from your LDAP.

```
#####
## USER Property for CP4BA ##
#####
## Use this parameter to specify the license for the CP4A deployment and
## the possible values are: non-production and production and if not set, the license
## will
## be defaulted to production. This value could be different from the other licenses in
## the CR.
CP4BA.CP4BA_LICENSE="<Required>"
## On OCP, the script populates these three (3) parameters based on your input for
## "production" deployment.
## The script populates the storage parameters based on your input.
## If you manually deploy without using the deployment script, then you must enter the
## different storage classes for the slow, medium,
## and fast storage parameters. If you only have 1 storage class defined, then you can
## use that 1 storage class for all 3 parameters.
## The sc_block_storage_classname is for Zen. Zen requires block storage (RW0) for the
## metastore DB.
CP4BA.SLOW_FILE_STORAGE_CLASSNAME="<my_file_classname>"
CP4BA.MEDIUM_FILE_STORAGE_CLASSNAME="<my_file_classname>"
CP4BA.FAST_FILE_STORAGE_CLASSNAME="<my_file_classname>"
CP4BA.BLOCK_STORAGE_CLASS_NAME="<my_block_classname>"

## Enable/disable FIPS mode for the deployment (default value is "false").
CP4BA.ENABLE_FIPS="false"

## Enable or disable the generation of network policies.
CP4BA.ENABLE_GENERATE_SAMPLE_NETWORK_POLICIES="true"

#####
## USER Property for BAN ##
#####
## Provide the user name for BAN. For example: "BANAdmin"
BAN.APLOGIN_USER="<Required>"
## Provide the user password for BAN.
```



```
BAN.APLOGIN_PASSWORD=<Required>"
## Provide LTPA key password for BAN deployment.
BAN.LTPA_PASSWORD=<Required>"
## Provide keystore password for BAN deployment.
BAN.KEYSTORE_PASSWORD=<Required>"
## Provide the user name for jMail used by BAN. For example: "jMailAdmin"
BAN.JMAIL_USER_NAME=<Optional>"
## Provide the user password for jMail used by BAN.
BAN.JMAIL_USER_PASSWORD=<Optional>"
```

Some values like a connection point name and table spaces for the Workflow object store initialization of Content can be any string, but are needed for the deployment to identify them.

**Important:** The passwords that are used for `CONTENT.LTPA_PASSWORD` and `BAN.LTPA_PASSWORD` must be the same in the `cp4ba_user_profile.property` file.

- b. Follow the instructions in the `cp4ba_user_profile.property` file to enter all the `<Required>` values for the external PostgreSQL DB.

```
## Please get "<your-server-certification: root.crt>" "<your-client-certification:
client.crt>" "<your-client-key: client.key>"
from server and client, and copy into this directory.
Default value is "/home/cert-kubernetes/scripts/cp4ba-prerequisites/project/
<CP4BA_NAMESPACE>/propertyfile/cert/zen_external_db".
CP4BA.ZEN_EXTERNAL_POSTGRES_DATABASE_SSL_CERT_FILE_FOLDER="/home/cert-kubernetes/scripts/
cp4ba-prerequisites/project/<CP4BA_NAMESPACE>/propertyfile/cert/zen_external_db"
## Name of the schema to store monitoring data. The default value is "watchdog".
CP4BA.ZEN_EXTERNAL_POSTGRES_DATABASE_MONITORING_SCHEMA="watchdog"
## Name of the database. The default value is "zencnpdb".
CP4BA.ZEN_EXTERNAL_POSTGRES_DATABASE_NAME="zencnpdb"
## Database port number. The default value is "5432".
CP4BA.ZEN_EXTERNAL_POSTGRES_DATABASE_PORT="5432"
## Name of the read database host cloud-native-postgresql on k8s provides this endpoint.
If DB is not running on k8s then same hostname as DB host.
CP4BA.ZEN_EXTERNAL_POSTGRES_DATABASE_R_ENDPOINT=<Required>"
## Name of the database host.
CP4BA.ZEN_EXTERNAL_POSTGRES_DATABASE_RW_ENDPOINT=<Required>"
## Name of the schema to store zen metadata. The default value is "public".
CP4BA.ZEN_EXTERNAL_POSTGRES_DATABASE_SCHEMA="public"
## Name of the database user. The default value is "zencnp_user".
CP4BA.ZEN_EXTERNAL_POSTGRES_DATABASE_USER="zencnp_user"
```

- c. If you entered a single database server, then the `DB_ALIAS_NAME` is set automatically. If you need more than one server, edit the `<DB_ALIAS_NAME>` property prefixes in the `cp4ba_db_name_user.property` file to assign each component to a database server name defined in the `DB_SERVER_LIST` key name in the `cp4ba_db_server.property` file. The value of a `<DB_ALIAS_NAME>` in the username property file must match a value that is defined in the `DB_SERVER_LIST`.

**Note:** If you see a section that is commented out with a single hash (`#`), verify that the capability is enabled and then uncomment these parameter lines. Enter the correct values for the username and password.

The following example shows the commented lines for the Case History database parameters.

```
## Provide the name of the database for Case History when Case History Emitter is
enabled. For example: "CHOS"
# <DB_ALIAS_NAME>.CHOS_DB_NAME="CHOS"
## Provide database schema name. This parameter is optional. If not set, the schema name
is the same as database user name.
## For DB2, the schema name is case-sensitive, and must be specified in uppercase
characters.
# <DB_ALIAS_NAME>.CHOS_DB_CURRENT_SCHEMA=<Optional>"
## Provide the user name for the object store database required by Case History when
Case History Emitter is enabled. For example: "dbuser1"
# <DB_ALIAS_NAME>.CHOS_DB_USER_NAME=<youruser1>"
## Provide the password (if password has special characters then Base64 encoded with
{Base64} prefix, otherwise use plain text) for the user of Object Store of P8Domain.
# <DB_ALIAS_NAME>.CHOS_DB_USER_PASSWORD="{Base64}<yourpassword>"
```

- d. Enter the required values for the LDAP variables in the `cp4ba_LDAP.property` file.

Replace the `<Required>` string with your existing LDAP server parameters, its query objects, users, and groups.

**Important:** If your target platform is ROKS Virtual Private Cloud (VPC), you can validate the connection to your LDAP only by using a VM client of the ROKS VPC. Set the LDAP server to the internal IP address or DNS of the ROKS VPC. For example, if the IP address of your LDAP is 10.240.0.16, then change the LDAP\_SERVER property in the cp4ba\_LDAP.property file to this address.

```
## The name of the LDAP server to connect
LDAP_SERVER="10.240.0.16"
```

If your client is not connected to the ROKS VPC, you can still set the IP address to propagate the value to the custom resource.

- e. All names and passwords in the cp4ba\_db\_name\_user.property file must be entered manually.

Replace the <yourpassword> strings with your database user passwords.

**Restriction:** The username values cannot contain special characters. Special characters include the equal sign (=), a forward slash (/), a colon (:), a single dot (.), single quotation marks ('), and double quotation marks ("). If a value does contain a special character, the script fails to parse the value.

The password can contain special characters, except the single quotation mark ('). The single quotation is used to enclose the string that contains special characters. If you have a password without special characters use the string as plain text. For example, if you want the password to be mypassword, specify the password as "mypassword".

```
dbserver1.GCD_DB_USER_NAME="GCDDDB"
dbserver1.GCD_DB_USER_PASSWORD="mypassword"
```

If you have a password with special characters (&passw0rd), you must encode this string and add {Base64} before you add the value to the property. To encode the password on Linux, run the following command:

```
# echo -n '&passw0rd' | base64
JnBhc3N3MHJk
```

Add the encoded value to the property with the {Base64} prefix.

```
dbserver1.GCD_DB_USER_NAME="GCDDDB"
dbserver1.GCD_DB_USER_PASSWORD="{Base64}JnBhc3N3MHJk"
```

6. When the user property files are complete and ready, make sure that you are in the scripts folder under cert-kubernetes, and run the cp4a-prerequisites.sh script in the "generate" mode.

```
./cp4a-prerequisites.sh -m generate -n $NAMESPACE
```

**Note:** If the script detects that the property files do not have custom values, the script stops and displays messages to help identify the missing values:

```
Change the prefix "<DB_ALIAS_NAME>" in propertyfile/cp4ba_db_name_user.property to assign
which database is used by the component.
Found invalid value(s) "<Required>" in property file "propertyfile/
cp4ba_db_name_user.property". Enter the correct value.
```

The following messages are displayed at the end of the execution:

```
[INFO] The DB SQL statement files for CP4BA are in
directory <extracted_cert-kubernetes_archive>/cert-kubernetes/scripts/cp4ba-prerequisites/
project/<CP4BA_NAMESPACE>/dbscript, you can modify or use the default settings to create
the database. (DO NOT CHANGE DBNAME/DBUSER/DBPASSWORD DIRECTLY)
[NEXT ACTIONS]
Enter the correct values in the YAML templates for
the secrets under <extracted_cert-kubernetes_archive>/cert-kubernetes/scripts/cp4ba-
prerequisites/project/<CP4BA_NAMESPACE>/secret_template
...
```

The `/cp4ba-prerequisites/project/<CP4BA_NAMESPACE>` directory has the following structure and varies depending on the capabilities that you selected when you ran the script:

```
├─ create_secret.sh
├─ dbscript
│   └─ <component>
│       └─ <database_type>
│           └─ <db_server_alias>
│               └─ <sql_template>
├─ propertyfile
├─ secret_template
│   └─ <component>
│       └─ <yaml_secret>
└─ ibm-ldap-bind-secret.yaml
```

If you chose an external PostgreSQL for Zen and IM in the property mode, then the generate mode creates the following files under the `propertyfile` folder.

- `zen_external_db/ibm-zen-metastore-edb-cm.yaml`: Use the generated YAML file to create a configMap.
  - `im_external_db/ibm-im-metastore-edb-cm.yaml`: Use the generated YAML file to create a configMap.
  - `zen_external_db/ibm-zen-metastore-edb-secret.sh`: Use the generated script to create the secret.
  - `im_external_db/ibm-im-metastore-edb-secret.sh`: Use the generated script to create the secret.
7. Check that you have all the necessary files for your CP4BA deployment. Copy the required database and LDAP certificates into the target directories as indicated by the `NEXT ACTIONS` messages from the script. Make sure that the scripts and the YAML files have the correct values.

For more information, see [Importing the certificate of an external service](#).

8. **If required:** If you selected a database type other than EDB Postgres, you or a database administrator must run the DB scripts against your database servers and use the YAML files to create the necessary secrets in your OpenShift Container Platform cluster.

**Remember:** The target databases must meet the requirements of the capabilities that you want to install. Review the preparing sections for each capability before you go ahead and create the databases. For more information, see [Preparing your chosen capabilities](#).

For example, consider your intended workload when you configure the database services. If your deployment includes FileNet® Content Manager, then review the important configuration tasks for the external database services under [Preparing the databases](#).

- For an external PostgreSQL database, the `psql` command line interface can be used to run the SQL scripts and manage the database configuration. The databases can be created by running the following command with your PostgreSQL instance user.

```
psql -U "$DB_USER" -f "$filename"
```

`$DB_USER` is the PostgreSQL user, and `$filename` is the path to your SQL file.

- If you chose the Db2® type for your database, run the SQL files to create your database by using the `db2` command line interface. The `$filename` is the path to your SQL file.

```
db2 -tvf "$filename"
```

- Microsoft SQL Server (MSSQL) provides a more interactive UI, and therefore, you can open the SQL files with SQL Server Studio on your MSSQL machine, and then run it by clicking **execute** to create the databases.

**Tip:** If you configured your deployment to use RDS Db2 (`dc_database_type=db2rds` or `dc_database_type=db2rdsHADR`), the generated SQL script must be run as a stored procedure with administrator user credentials to create the database. Leave enough time in between the SQL

statements for previous commands to complete. A few minutes are often needed to create the RDS Db2 database.

```
CALL rdsadmin.create_database('GCDDDB',32768,'UTF-8','US');
```

After the database is created, you can then run further stored procedures if you want. For example, stored procedures can be used to create the buffer pool, table spaces, and users.

```
-- Create buffer pool
CALL rdsadmin.create_bufferpool('GCDDDB','GCDDDB_1_32K',1024,'Y','Y',32768,0,32);
CALL rdsadmin.create_bufferpool('GCDDDB','GCDDDB_2_32K',1024,'Y','Y',32768,0,32);
-- Create table spaces
CALL
rdsadmin.create_tablespace('GCDDDB','GCDDATA_TS','GCDDDB_1_32K',32768,NULL,NULL,'U','AUTOMATIC
');
CALL
rdsadmin.create_tablespace('GCDDDB','GCDDDB_TMP_TBS','GCDDDB_2_32K',32768,NULL,NULL,'T','AUTOMA
TIC');
-- Create user
CALL rdsadmin.create_role('GCDDDB','FNCM');
CALL rdsadmin.add_user('gcduser','smartpasswordgoeshere',null);
CALL rdsadmin.grant_role('GCDDDB','FNCM','USER gcduser','N');
CALL rdsadmin.update_db_param('GCDDDB','LOCKTIMEOUT','30');
```

If you create users, run the GRANT permissions statement for all the users of the database. In the following example, the db admin must connect to the database (GCDDDB) to run the SQL statements.

```
-- Grant permissions to DB user
GRANT CREATETAB,CONNECT ON DATABASE TO USER gcduser;
GRANT USE OF TABLESPACE GCDDATA_TS TO USER gcduser;
GRANT USE OF TABLESPACE GCDDDB_TMP_TBS TO USER gcduser;
GRANT SELECT ON SYSIBM.SYSVERSIONS TO USER gcduser;
GRANT SELECT ON SYSCAT.DATATYPES TO USER gcduser;
GRANT SELECT ON SYSCAT.INDEXES TO USER gcduser;
GRANT SELECT ON SYSIBM.SYSDUMMY1 TO USER gcduser;
GRANT USAGE ON WORKLOAD SYSDEFAULTUSERWORKLOAD TO USER gcduser;
GRANT IMPLICIT_SCHEMA ON DATABASE TO USER gcduser;
-- GRANT USAGE ON SCHEMA RDSADMIN TO gcduser;
GRANT SQLADM ON DATABASE TO USER gcduser;
GRANT EXECUTE ON PACKAGE NULLID.SYSSN200 TO USER gcduser;
GRANT EXECUTE ON PACKAGE NULLID.SYSSH200 TO USER gcduser;
```

For more information about RDS Db2, see [Amazon RDS for Db2](#). For more information about stored procedures, see [Amazon RDS for Db2 stored procedure reference](#).

When you set up an RDS Db2 instance for your Cloud Pak for Business Automation deployments, use an Amazon Route 53 to configure a custom DNS name. A Route 53 creates a consistent endpoint for your CP4BA data source configurations, and introduces stability across database operations. For more information, see [AWS Route 53 Developer Guide](#).

- a. Run the following command to go to the target CP4BA namespace.

```
oc project $NAMESPACE
```

- b. Make sure that you are in the scripts folder under cert-kubernetes and run the create\_secret.sh script to apply all the YAML template files that you generated.

**Important:** The generated secrets include data and stringData sections in the YAML files. The data sections require values to be Base64 encoded. The stringData sections require values in plain text. If you need to modify any password in the secret template files before you create the secret in your cluster, then make sure that the updated fields are in the right format.

- If an updated value is in the data: section of a file, it must be Base64 encoded. A Base64 encoded value can be retrieved by running the following command.

```
echo "<value-to-update>" | base64
```

- If an updated value is in the `stringData:` section of a file, it can be in plain text.

```
./cp4ba-prerequisites/project/$NAMESPACE/create_secret.sh
```

- Optional: Before you validate your database and LDAP connections, you can set values for your language and country. If no values are set, English (`-Duser.language=en`) is set as the language, and the United States (`-Duser.country=US`) is set as the country.

Run the `export` command to set the values for a language and country as environment variables before you run the `cp4a-prerequisites.sh` script in the "validate" mode. The following variables set the language to Hindi and the country to India.

```
export CP4BA_AUTO_LANGUAGE="HI"
export CP4BA_AUTO_REGION="IN"
```

- Optional: When all the required databases and secrets are created, make sure that you are in the `scripts` folder under `cert-kubernetes`, and run the `cp4a-prerequisites.sh` script again in the "validate" mode.

```
./cp4a-prerequisites.sh -m validate -n $NAMESPACE
```

The command validates that the storage classes that you entered in the property mode meet the RWX and RWO requirements. If the validation is successful, it is marked as **PASSED!**

The command also checks that the required secrets are found and submits a validation query to the LDAP server and the list of remote database servers. If you chose an external Postgres DB used for the Zen metastore, then the connection is also checked. If the operations succeed within the timeout threshold, the validation is marked as **PASSED!** No queries are run and no data is changed, the script just reports that the connection succeeded.

If a connection is not successful, then a message informs you which connection failed. To resolve the issue, check the values in your property files so that you can correct them and try again.

**Note:** The `cp4a-prerequisites.sh -m validate` command uses a simple JDBC method to test the connection to the remote database servers with the Cloud Pak for Business Automation default JDBC drivers.

If you need to use customized JDBC drivers in your production deployment, you can locate these drivers by using the `sc_drivers_url` parameter during the configuration of the custom resource. For more information, see [Optional: Preparing customized versions of JDBC drivers and ICCSAP libraries](#).

## Results

**Tip:** You can change (add or remove) the selected CP4BA capabilities that you want to prepare for by rerunning the script and merging the new property files with the backed-up property files.

You can rerun the script in the "property" mode to create new property files. When the script detects it ran before, the previous property folder is renamed into a new time-stamped folder. The name of the backed-up folder is `cert-kubernetes/scripts/cp4ba-prerequisites-backup/project/$NAMESPACE/propertyfile_%Y-%m-%d-%H:%M:%S`.

Use the following steps to update your property files to include your updated capabilities:

- Copy the file `.tmp/.TEMPORARY.property` into a back up file, for example `.TEMPORARY.property.backup`.
- Rerun the `cp4a-prerequisites.sh` script in the "property" mode, and choose a different selection of capabilities.
- Restore the `cp4ba_LDAP.property` and `cp4ba_External_LDAP.property` files from the backup folder by copying and pasting them into the new folder.
- Compare the `cp4ba_db_server.property` file from the backup folder and merge it where necessary with the new `cp4ba_db_server.property` file.

5. Merge the new `cp4ba_db_name_user.property` and `cp4ba_user_profile.property` files with the backed-up property files.
6. Rerun the `cp4a-prerequisites.sh` script in the "generate" mode to update the database SQL statements and YAML templates for the secrets.
7. Compare and merge the `.TEMPORARY.property.backup` file with the `.tmp/.TEMPORARY.property` file for the new capabilities.
8. Run the database SQL statements for the new capabilities.
9. Create the secrets for the new capabilities.

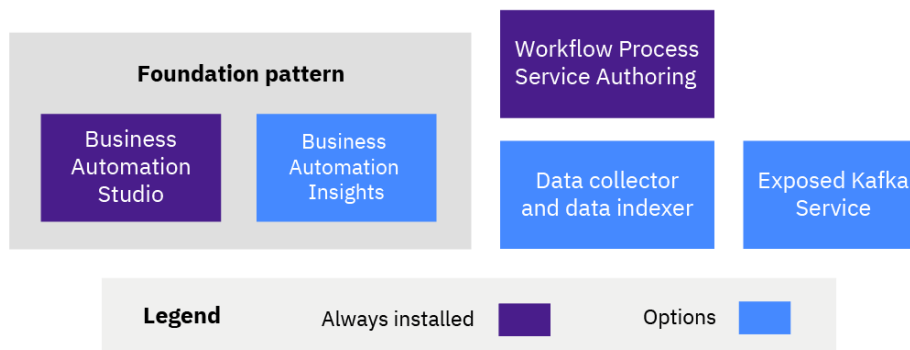
If you already installed a CP4BA deployment and want to update it with the new databases and secrets for the new capabilities, you must run the `cp4a-deployment.sh` again to update the custom resource. Do not forget to verify the custom resource YAML before you scale down the deployment, apply the new custom resource with the `--overwrite=true` parameter, and scale the deployment back up. For more information, see [Applying the upgraded custom resource](#).

## Preparing to install Workflow Process Service Authoring

Before you install Workflow Process Service Authoring, you must prepare your environment.

### Before you begin

The following diagram shows the components that you can select for Workflow Process Service Authoring. You must make sure that you prepare everything that is needed by these components before you apply your custom resource.



If you used the `cp4a-prerequisites.sh` script to generate the database SQL statement files (scripts) and YAML template files for the database secrets, then you do not need to create these resources and apply them again. For more information about running the `cp4a-prerequisites.sh` script, see [“Preparing databases and secrets for your chosen capabilities by running a script”](#) on page 1.

**Tip:** When you use the `cp4a-prerequisites.sh` script, you must review the requirements in the documented steps to create a database. Make sure that the databases that you create satisfy your intended workload. For deployments that need to operate continuously with no interruptions in service, set up a high availability (HA) database.

### Procedure

1. Optional: You can create your databases and secrets without running the provided scripts.

**Restriction:** If you previously installed another pattern that includes Business Automation Studio, you do not need to configure a new database for Business Automation Studio. Otherwise, PostgreSQL is the only database supported for Workflow Process Service Authoring.

By default, EDB Postgres will be provisioned for Workflow Process Service Authoring. If you want to manually configure your own PostgreSQL database, you can use the script in step 2 and 3 in [Creating PostgreSQL databases](#).

2. By default, dynamic provisioning is enabled when you deploy. If your environment supports dynamic provisioning, you can skip the rest of step 2 and step 3. If your environment does not support dynamic provisioning, complete the following steps to create persistent volumes (PVs) manually.

To enable persistent storage for the server logs, dump files, or index files for Business Automation Studio, you can either use dynamic provisioning that you already set up in your cluster, or you can create the PV manually.

Disable dynamic provisioning by setting `bastudio_configuration.storage.enabled` to `false`. Then, create the required PVs and persistent volume claims (PVCs) manually and set `bastudio_configuration.storage.existing_pvc_for_logstore` to the name of the log PVC and `bastudio_configuration.storage.existing_pvc_for_dumpstore` to the name of the dump PVC. The persistent storage configuration of index files is different from that of server logs and dump files.

3. Create a PV for the Workflow Process Service Authoring file store. You have the following options, depending on whether your Kubernetes environment supports dynamic provisioning.
  - Option 1: If your environment supports dynamic provisioning:  
Enable dynamic provisioning by setting `workflow_authoring_configuration.storage.use_dynamic_provisioning` to `true`.
  - Option 2: If your environment does not support dynamic provisioning:  
Disable dynamic provisioning by setting `workflow_authoring_configuration.storage.use_dynamic_provisioning` to `false`. Then, create the required PV and PVC manually and set `workflow_authoring_configuration.storage.existing_pvc_for_filestore` to the name of the generic file PVC.
4. If you selected the optional IBM Business Automation Insights component, complete the following steps to configure it:
  - a) Make sure that you complete all the steps in [Preparing to install IBM Business Automation Insights](#).
  - b) Optional: If you want Business Automation Insights to capture events from BPMN processes for further processing, install and configure the BPM event emitter application for Workflow Process Service Authoring. For more information, see [Installing and configuring the BPM event emitter](#).

## Creating a production deployment

You can deploy a custom resource file from the OpenShift console, or you can create a custom resource file by running the deployment script.

### About this task

The following progress bar shows you where you are in the installation process.



You have two options to create the custom resource and install a CP4BA deployment. Option 1 is the recommended option.

#### Option 1: Running the deployment script

1. Generating the custom resource with the deployment script.
2. Checking and completing your custom resource.

3. Deploying the custom resource that you created with the deployment script.
4. Validating your production deployment.
5. Optional: Installing network policies.

#### Option 2: Using the OpenShift console

1. Installing a production deployment in the OpenShift console.
  - Checking and completing your custom resource.
2. Validating your production deployment.
3. Optional: Installing network policies.

## Generating the custom resource with the deployment script

Depending on the capabilities that you want to install, the deployment script generates a custom resource file with the selected automation containers.

### About this task

The script creates a custom resource (CR) file to deploy by the Cloud Pak operator. The deployment script prompts the user to enter values to get access to the container images and to select what is installed with the deployment.

**Note:** The deployment script uses a custom resource (CR) template file for each pattern. The pattern template names include "production" and are found in the `cert-kubernetes/descriptors/patterns` folder. The CR files are configured by the deployment script.

**Remember:** You can run the scripts on an amd64/x86, a Linux on Z, or a Linux on Power® based cluster.

### Procedure

1. Log in to the cluster with the cluster administrator that you used in [Preparing for a production deployment](#) or a non-administrator user who has access to the project.

Using the Red Hat OpenShift CLI:

```
oc login https://<cluster-ip>:<port> -u <cluster-admin> -p <password>
```

2. View the list of projects in your cluster to see the target project before you run the deployment script.

```
oc get projects
```

**Note:** If you used the `All namespaces` option to install the Cloud Pak operator, then you must have another project in addition to `openshift-operators` in the cluster before you create the deployment.

3. Change the scope to the project that you created for your deployment (`cp4ba-project`). If you chose to separate duties of the CP4BA operators and operands, use the project that you set for the `$CP4A_OPERAND_NAMESPACE`.

```
export NAMESPACE=<project_name>
oc project ${NAMESPACE}
```

The specified project is used in all subsequent operations that manipulate project-scoped content.

4. If you need to, download the `cert-kubernetes` repository to a Linux based VM/machine.

For more information about downloading `cert-kubernetes`, see [Preparing for a production deployment](#).

5. Run the deployment script from the local directory where you downloaded the `cert-kubernetes` repository, and follow the prompts in the command window.

**Note:**



Multiple capabilities can be selected and deployed in the same namespace. The script proposes new options each time you select a capability. Some capabilities are mutually exclusive, so these combinations are never proposed. For more information about the capabilities and their dependencies, see [Capability patterns for production deployments](#).

If you select Workflow Runtime and Workstreams, two server instances are created, one server for each capability. If you want to install both capabilities into one server instance, you need to create the CR by hand and use the fully customizable (FC) template (`ibm_cp4a_cr_production_FC_workflow-workstreams.yaml`).



**Warning:** The script deletes any previously generated custom resource files under the `cert-kubernetes/scripts/generated-cr` folder. If you want to keep them, make copies of the YAML files somewhere else.

```
cd cert-kubernetes/scripts
./cp4a-deployment.sh -n ${NAMESPACE}
```

- a. Accept the license. You must agree to the license that is displayed.
- b. If you already deployed a CP4BA FileNet Content Manager instance in your namespace, then select Yes. The default is No.
- c. Select the Production deployment type.

The `cp4a-deployment.sh` script checks your local file system for the property files that are created by the `cp4a-prerequisites.sh` script. If the following files are found, then the script assumes that you ran the `cp4a-prerequisites.sh` script, generated the database scripts and secrets, and validated that the property values that you entered are correct for your deployment.

```
cert-kubernetes/scripts/.tmp/.TEMPORARY.property
cert-kubernetes/scripts/cp4ba-prerequisites/project/${NAMESPACE}/propertyfile/
cp4ba_db_name_user.property
cert-kubernetes/scripts/cp4ba-prerequisites/project/${NAMESPACE}/propertyfile/
cp4ba_db_server.property
cert-kubernetes/scripts/cp4ba-prerequisites/project/${NAMESPACE}/propertyfile/
cp4ba_LDAP.property
cert-kubernetes/scripts/cp4ba-prerequisites/project/${NAMESPACE}/propertyfile/
cp4ba_user_profile.property
```

- d. Select the platform type: ROKS (1) or OCP (2).

If you selected OpenShift Container Platform (OCP), then choose one of the following options:

- If your OCP is deployed on AWS or Azure, select Yes.
  - If your OCP is not deployed on AWS or Azure, then select No.
- e. If you need to, provide a URL to the `.zip` file that contains your custom JDBC or IBM® Content Collector for SAP Applications drivers.

For more information, see [Optional: Preparing customized versions of JDBC drivers](#).

- f. If you plan to have restricted network policies, and you want the operators to generate sample network policies templates, select Yes (Yes/No. Default: No). The CP4BA operator is no longer creating network policies automatically.

**Important:** If you have any existing `deny-all` network policies in your CP4BA namespace, you must remove them before you apply the CP4BA CR. You can recreate them after the installation by running the `cp4a-network-policies.sh` script to copy the network policies templates from the operators to a local location.

For more information, see [Configuring cluster security](#).

- g. A summary of your selection is displayed. Click "Yes" to verify that the information is correct.

## Results

A custom resource file is created `generated-cr/project/${NAMESPACE}/ibm_cp4a_cr_final.yaml`.

**Tip:** You can rename the file and move it to another folder, or you can continue to use the file from this location. Because the maximum length of labels in Kubernetes is 63 characters, be careful with the lengths of your CR name and instance names. Some components can configure multiple instances. Each instance must have a different name. The total length of the CR name and an instance name must not exceed 24 characters, otherwise some component deployments fail. When your deployment contains Workflow Authoring, the total length of the CR name cannot exceed 22 characters.

## Checking and completing your custom resource

A custom resource YAML is a configuration file that describes an instance of a deployment that is created by one or more of the CP4BA operators and includes parameters to install some or all of the Cloud Pak capabilities.

### About this task

A single custom resource file is used to include all of the capabilities that you want to install with a particular operator. Each time that you need to make an update or modification, you must apply the changes to your deployments. When you apply a new custom resource to an operator, you must make sure that all previously deployed resources are included, otherwise the operator deletes them.

**Note:** Do not modify the **appVersion** parameter when you configure the YAML file. Change the parameter only if you upgrade from a previous version.

## Checking the cluster configuration

You must check and edit the shared sections of the compiled custom resource file before you apply it to the operator.

### About this task

In all cases, check the <Required> values for the **image\_pull\_secrets** and **images** parameters in the **shared\_configuration** section. For more information, see [Shared configuration parameters](#).

Table 1. Checking <Required> parameters for selected images	
Parameter	Description
dbcompatibility_init_container	Repository from where to pull the Application Engine init_container and the corresponding tag.
image_pull_secrets	Secrets in your target namespace to pull images from the specified repository.

### Procedure

1. Locate the **shared\_configuration** section in the custom resource (CR) file that you created in “Generating the custom resource with the deployment script” on page 14, then check and correct the deployment parameters.

The custom resource templates can include the following parameters:

#### License parameters

- **sc\_deployment\_license**, which can be non-production, or production.
- **sc\_deployment\_fnm\_license**, which can be: user, concurrent-user, authorized-user, non-production, or production.
- **sc\_deployment\_baw\_license**, which can be: user, non-production, or production.

**Note:** You can ignore the fnm and baw parameters if they do not appear in your custom resource. For more information, see [Shared configuration](#).

## Platform parameters

- **sc\_deployment\_platform**, which can be "OCP" or "ROKS".
- **sc\_ingress\_enable** must be set to `true` to create an ingress on ROKS.

## Sizing parameters

**sc\_deployment\_profile\_size**, which determines the profile of your deployment. The default is `small`, but you can change the profile to `medium` or `large`.

## Huge pages

**sc\_hugepages**, enables the allocation of huge pages that are configured in the cluster to your CP4BA deployment. By default, huge pages allocation is not enabled. If you want your CP4BA deployment to use the huge pages that you enabled on the cluster, then set **sc\_hugepages.enabled** to `true`. For more information, see [Shared configuration](#).

## Storage parameters

These parameters are mandatory.

- **sc\_slow\_file\_storage\_classname**
- **sc\_medium\_file\_storage\_classname**
- **sc\_fast\_file\_storage\_classname**
- **sc\_block\_storage\_classname**

2. Optional: Configure the root secret, external SSL/TLS certificate secret, and the trusted certificate list.

The custom YAML file includes the **root\_ca\_secret**, **external\_tls\_certificate\_secret**, and **trusted\_certificate\_list** parameters. The **root\_ca\_secret** parameter is the name of the secret that contains the root CA signer certificate for the Cloud Pak. If the secret does not exist, then a self-signed signer certificate is generated. For more information, see [Changing the default root CA signer certificate](#).

For production environments, it is likely that you want to use your own certificates that are trusted by your clients. The **external\_tls\_certificate\_secret** parameter is used to store a wildcard certificate, which can be more convenient than a certificate for each subdomain. A multi-domain wildcard certificate can also be used to secure multiple domains and their subdomain names. For more information, see [Creating secure endpoints for external services](#).

**Important:** If you choose to use self-signed certificates, certain features of the product might not work as expected because of modern browser restrictions that are related to self-signed certificates. A browser blocks any redirect to a site that uses a certificate that is not signed by a root CA that is trusted by the browser. This can result in access issues for business applications.

The **trusted\_certificate\_list** parameter can be used to trust root CA certificates for external services. For more information, see [Connecting endpoints to external services over TLS](#).

3. Check the **resource\_registry\_configuration** section.

Automatic backup for the Resource Registry is recommended. For more information, see [Enabling Resource Registry disaster recovery](#).

4. Check the values for the **image\_pull\_secrets** parameter, the **sc\_image\_repository** parameter, and for the Application Engine repositories.

All components use the same docker image repository. By default, the IBM Entitlement Registry is used `cp.icr.io`.

```
shared_configuration:
  sc_image_repository: cp.icr.io
  image_pull_secrets:
    - ibm-entitlement-key
```

For an air gap installation, make sure that the **sc\_image\_repository** parameter is set to the default value. The `images` section is needed only if your environment is offline (air gapped deployment).

```
shared_configuration:
  sc_image_repository: cp.icr.io
```

```

image_pull_secrets:
- ibm-entitlement-key
images:
  dbcompatibility_init_container:
    repository: <registry_url>:5000/<namespace>/dba-dbcompatibility-initcontainer
    tag: <version>
    pull_policy: IfNotPresent

```

The `<version>` number is 25.0.0.

**Note:** The `images` section is needed only if your environment is offline (air gapped deployment).

5. **If required:** If you did not run the `cp4a-prerequisites.sh` script and you did not select EDB Postgres as the only database type, then enter the parameter values for your data source instance in the **`datasource_configuration`** section.

Your deployment might need several databases. Follow the configuring instructions for each component to complete this section.

6. Optional: If you selected EDB Postgres as a database type for any of the capabilities, check the database parameters under the **`datasource_configuration`** section as they are used if present in the custom resource.

The value of the **`database_name`** is used to create the database for that component. The **`database_ssl_secret_name`** is set to the secret that is created for the `{{ meta.name }}-pg-client-cert-secret` parameter.

The following fields are mandatory:

- GCD datasource configuration: Enter a valid database name.
- Navigator datasource configuration: Enter a valid database name.
- Content Management OS datasource configuration: Enter a valid database name and database server name, for example `postgres-cp4ba-1w.{{ meta.namespace }}.svc`.

The following database names are used for each component if these parameters are not specified in the custom resource.

Table 2. Component database default names	
Components	Default database names
<code>dc_ads_designer_datasource</code>	<code>adsdesignerdb</code>
<code>dc_ads_runtime_datasource</code>	<code>adsruntimedb</code>
<code>dc_gcd_datasource</code>	<code>gcdcdb</code>
<code>dc_os_datasources</code>	<code>os + (index) + db</code>
<code>dc_cpe_datasources</code>	<code>chos + (index)</code>
<code>dc_icn_datasource</code>	<code>icndb</code>
<code>dc_odm_datasourcure</code>	<code>odmdb</code>
<code>dc_adp_datasource</code>	<code>adpggdb</code>
<code>bastudio_configuration</code>	<code>basdb</code>
<code>bastudio_configuration.playback_server.database.name</code>	<code>appdb</code>
<code>baw_configuration</code> <ul style="list-style-type: none"> <li>• document object store database</li> <li>• design object store database</li> <li>• target object store database</li> </ul>	<code>bawdb + (index)</code> <ul style="list-style-type: none"> <li>• <code>bawdocs</code></li> <li>• <code>bawdos</code></li> <li>• <code>bawtos</code></li> </ul>

Table 2. Component database default names (continued)	
Components	Default database names
application_engine_configuration	aaedb + (index)

- Optional: Modify the default value (cpadmin) of the **sc\_iam.default\_admin\_username** parameter for the Identity Management (IM) foundational service.

The IM admin username cannot be the same as a user in your LDAP. If your LDAP has a user with the name cpadmin, then set a different default admin username for IM.

- Optional: By default, all the CP4A pods have unrestricted network access to external systems. Set the value of **sc\_generate\_sample\_network\_policies** to true to create sample network policies from CP4BA capability templates.

For more information, see [Shared configuration parameters](#).

For more information about customizing your network security, see [Configuring cluster security](#).

## Configuring Workflow Process Service Authoring

Before you install, configure the custom resource YAML file for Workflow Process Service Authoring.

### Before you begin

Follow the instructions in [Preparing Workflow Process Service Authoring](#).

### Procedure

- Open the CR file that you created in [“Generating the custom resource with the deployment script”](#) on page 14.
- Check the values to make sure that they are the values that you want to deploy.
  - Make sure that the `shared_configuration` section of the CR file contains values for all required parameters, such as `sc_deployment_license`.
  - LDAP configuration (`ldap_configuration`) is not required. If you want to use LDAP, you can add the related parameters into your CR file.
  - By default, EDB Postgres is provisioned.
    - If you ran the `cp4a-prerequisites.sh` script to prepare your databases and Kubernetes secrets, and you selected Document Processing with Workflow Process Service Authoring, then you must update the database section of `bastudio_configuration` to use PostgreSQL. You must create the PostgreSQL database for your deployment without the help of the `cp4a-prerequisites.sh` script. For more information, see [Creating PostgreSQL databases](#).
    - If you want to use your own PostgreSQL, you can add the configuration under the `bastudio_configuration` section. The following custom resource YAML shows the required parameters.

```

database:
  type: postgresql
  ## Provide the database server hostname for BASTudio
  host: "<Required>"
  ## Provide the database name for BASTudio. For example, wfpsdb. This parameter is
  case sensitive.
  ## The database provided should be created by the BASTudio SQL script template.
  name: "<Required>"
  ## Provide the database server port for BASTudio
  port: "<Required>"
  # ## If you want to enable PostgreSQL connection fail-over, you must configure
  alternative_host and alternative_port. Otherwise, leave them blank.
  #   alternative_host:
  #   alternative_port:
  #   # ssl_enabled: <true/false>
  #   # # After you enable the database SSL connection with true, dave the TLS
  certificate used by the database in a secret

```

```
# # # and put the name here. The secret can be created with the command:
# # # kubectl create secret generic <db_certificate_secret_name> --from-
file=tls.crt=<certificate_pem_file_location>
# # certificate_secret_name: <db_certificate_secret_name>
```

For more information, see [Workflow Process Service Authoring parameters](#).

3. If you are using the default EDB Postgres installed with IBM Workflow Process Service Authoring, you need to self-manage this database. When you installed Workflow Process Service Authoring, the IBM Cloud Pak® for Business Automation operator setup the database with default values.
  - a) Optional: If you want to change the default configuration, update the custom resource Cluster/<your-cr-name>-wfps-db with your changes to the database configuration. For more information about updating the custom resource parameters and cluster resources, see [PostgreSQL Configuration](#) and [Resource management](#).
  - b) Optional: If you want to switch to an external database, update your ICP4ACluster custom resource's Business Automation Studio database section. When configuring, the database name is wpsdb and the user is wpsuser. You can find the information for all the other fields in the Cluster/<your-cr-name>-wfps-db status. For example, the database section might look similar to:

```
database:
  type: "PostgreSQL"
  name: "wpsdb"
  host: "<Your_CR_Name>-wfps-db-1w.<Your_Namespace>.svc"
  port: "5432"
  ssl_enabled: "true"
  certificate_secret_name: "<Your_CR_Name>-wfps-db-server"
  use_custom_jdbc_drivers: "false"
```

## Validating the YAML in your custom resource file

You must validate your custom resource (CR) file before you apply it. It is likely that you edited the file multiple times, and possibly introduced errors or missed values during your customizations.

### About this task

A good check before you apply the custom resource (CR) is to validate the YAML is executable. The web has many tools that you can use, but [YAML Lint](#) is simple and reliable.

If you want to use a command line tool, yq is a lightweight tool to help you check your YAML files. For more information, see [yq - portable yaml processor](#).

When yq is installed, you can run the following command to verify that the custom resource has no errors.

```
yq <custom-resource-file>.yaml
```



**Warning:** Your CR file might contain confidential and sensitive information. Remove all of your security parameters and their values before you paste the contents of the file into the **YAML Lint** tool.

### Procedure

1. Go to [YAML Lint](#).
2. Copy and paste the contents of your CR file (without your security parameters) into the gray text box, and click **Go**.

### Results

The tool reports whether the YAML is valid. If the YAML is valid, it strips out the comments and displays the configuration in a UTF-8 format. Skim the configuration again and check for any values that still show "<Required>". If you kept the comments in your CR, checking it without these lines can make it easier to read.

If the YAML is not valid, the tool generates a message and indicates the line where the error is found.

## What to do next

Correct any errors that are reported, and when your CR file has no errors make sure that the CR that you intend to use includes your security parameters.

## Deploying the custom resource you created with the deployment script

---

To install the deployment, you must apply the custom resource to the operator.

### Before you begin

Make sure that you followed the instructions to prepare your environment for all the capabilities you want to install, and you have access to all the container images. For more information, see [Getting access to images from the public IBM Entitled Registry](#).



**Warning:** If your target cluster is ROKS classic and the worker nodes rebooted, then you must synchronize the time on each of the worker nodes before you deploy the CP4BA custom resource. To synchronize the times on the worker nodes, run the following command from a connected client:

```
oc get no -l node-role.kubernetes.io/worker --no-headers -o name | xargs -I {}  
-- oc debug {} -- chroot /host sh -c 'systemctl restart chronyd'
```

### Procedure

1. Check that all the capabilities that you want to install are configured. If you selected CP4BA capabilities, the custom resource file is named `ibm_cp4a_cr_final.yaml`.

```
cat generated-cr/project/${NAMESPACE}/ibm_cp4a_cr_final.yaml
```

2. Use the OpenShift CLI to deploy the configured capabilities and apply the custom resource. For CP4BA capabilities, use the custom resource file `ibm_cp4a_cr_final.yaml`.

```
oc apply -f generated-cr/project/${NAMESPACE}/ibm_cp4a_cr_final.yaml
```

### Results

The operator reconciliation loop can take some time. You must verify that the automation containers are running.

When the deployment is complete, the message `Deployment created` is displayed.

1. You can open the operator log to view the progress. Using the OpenShift CLI:

```
oc logs <operator pod name> -c operator -n ${NAMESPACE}
```

Get the full syntax by entering the help command.

```
oc logs --help
```

2. Monitor the status of your pods from the command line. Using the OpenShift CLI:

```
oc get pods -w
```

3. When all the pods are "Running", you can access the status of your services with the following OCP CLI command.

```
oc status
```

See [Troubleshooting](#) to access the operator logs.

### **If EDB Postgres is used**

If you selected EDB Postgres as the database, the CP4BA operator creates an EDB Cluster customer resource for the EDB instance (postgres-cp4ba), and sets the authentication to the instance to use `sslmode=verify-ca`. The postgres-cp4ba instance hosts the CP4BA capabilities, and runs in a single pod (postgres-cp4ba-1). Multiple pods can be created by scaling up. A secret (postgres-cp4ba-app) is created that contains access information to the EDB cluster.

For each database that the operator creates in the EDB Postgres instance, it has a corresponding entry in the `pg_hba.conf` file for the database user to have access to it. For example, the user `gcdusr`, has access to the database `gcddb`.

The operator also generates the client certificate `{{ meta.name }}-pg-client-cert-secret` for the users to authenticate. The secret contains the following keys:

- `clientkey.pem`
- `clientcert.pem`
- `serverca.pem`
- `sslmode`

The EDB Postgres instance is not accessible outside of the cluster. To view the database in the EDB Postgres instance, use the following command.

```
oc rsh postgres-cp4ba-1 -n ${NAMESPACE}
```

From inside the EDB Postgres instance, you can run the `psql` command. By using the terminal-based front end to PostgreSQL you can type in queries interactively, issue them to PostgreSQL, and see the query results. For more information, see [psql](#).

```
psql -U postgres
```

## **What to do next**

When all the containers are running, you can access the services.

1. Go to the `cert-kubernetes` directory on your local machine.

```
cd cert-kubernetes
```

For more information about downloading `cert-kubernetes`, see [Preparing for a production deployment](#).

2. Log in to the cluster with the non-administrator user. Using the OpenShift CLI:

```
oc login
```

3. Look for the status field of each capability by running the `oc get` command.

```
oc get ICP4ACluster <instance_name> -o=jsonpath='{.status.components.<component_id>}'
```

Where the `<component_id>` can be any of the following ids:

```
status:
  components:
    ae-icp4adeploy-workspace-aae
    viewone
    gitgatewayService
    css
    contentDesignerRepoAPI
    adsLtpaCreation
    adsCredentialsService
    workflow-authoring
    graphql
    adsRrRegistration
```



```

adsRuntimeService
ae-icp4adeploy-pbk
app-engine
contentProjectDeploymentService
contentDesignerService
adsGitService
cmis
adsParsingService
bastudio
ier
adsRestApi
adsBuildService
navigator
baw
odm
cpe
iccsap
tm
adsFront
adsRunService
prereq
adsRuntimeBaiRegistration
resource-registry
pfs
adsDownloadService
ca
baml
extshare

```

4. Get the access information by running either of the following commands:

```
oc get cm <instance_name>-cp4ba-access-info -o=jsonpath='{.data.<component_id>-access-info}'
```

```
oc describe icp4acluster <instance_name> -n <namespace>
```

**Note:** The bastudio-access-info section provides access information for the Cloud Pak dashboard (Zen UI) and Business Automation Studio, which is installed by several patterns. The included URLs and credentials can be used to access the applications designers of the installed components.

Business Automation Studio uses the IBM Cloud Pak Platform UI (Zen UI) to provide a role-based user interface for all Cloud Pak capabilities. Capabilities are dynamically available in the UI based on the role of the user that logs in. The URL for the Admin Hub is included in the cp4ba-access-info ConfigMap.

You have two options to log in, **Enterprise LDAP** and **IBM provided credentials (cpadmin only)**. To log in to the Admin Hub to configure the LDAP, then click **IBM provided credentials (cpadmin only)**. You can get the details for the IBM-provided cpadmin user by getting the contents of the platform-auth-idp-credentials secret in the namespace used for the CP4BA deployment.

```
oc -n ${NAMESPACE} get secret platform-auth-idp-credentials -o jsonpath='{.data.admin_password}' | base64 -d && echo
```

If you want to log in using the configured LDAP, then click **Enterprise LDAP** and enter the cp4admin user and the password in the cp4ba-access-info ConfigMap. The cp4admin user has access to Business Automation Studio features.

If you want to add more users, you need to log in with the Zen UI administrator. The *kubeadmin* user in the Red Hat OpenShift authentication and the IBM-provided cpadmin user have the Zen UI administrator role.

You can change the default password at any time. For more information, see [Changing the cluster administrator password](#).

After you created a deployment, the operator automatically connects your LDAP to IM. The users and groups you defined in your LDAP are now available via IM.

You must associate your users and groups to Zen roles to be able to use them in all the applications. IBM Automation® has four roles that are defined: **Automation Administrator**, **Automation Analyst**, **Automation Developer**, and **Automation Operator**. For more information, see [Roles and permissions](#).

Log in to the [Common Web UI](#) to get the IBM Cloud Pak console route and admin's password. Use the [Platform UI \(Zen\)](#) to create a group for your *CP4BA Developers*, and add your LDAP users and groups to this group. You then need to assign the Zen group with the **Automation Developer** role.

**Note:** If you included multiple capabilities from FileNet Content Manager (FNCM), Automation Document Processing (ADP), and Business Automation Application (BAA) in your CP4BA deployment, then use the **Navigator for CP4BA** heading in the `cp4ba-access-info` ConfigMap and the custom resource status fields to find the route URL for Business Automation Navigator.

## Optional: Installing network policies

The CP4BA operators do not create network policies, but you can use the utility script `cp4a-network-policies.sh` to retrieve and install network policy templates to restrict network access.

### Before you begin

As a prerequisite to use the `cp4a-network-policies.sh` script, you must select the option to generate the network policy templates during the installation. If you want to generate the network policy templates after the installation, you can set the customer source parameter to true.

```
shared_configuration.sc_generate_sample_network_policies: true
```

If you chose to generate network policies, then wait for all the operators to complete the generation of the sample files in the `/tmp/<ns>/network-policies/templates` folder inside the CP4BA operator pod. You can check the status in the CP4BA custom resource (CR) to make sure that the deployment is complete (Ready).



**Attention:** The network policy templates might not work in all environments. You might need to adjust ingress or egress rules to make them work in your environment. For more information, see [Configuring cluster security](#).

### About this task

The `cp4a-network-policies.sh` script can be found in the `cert-kubernetes` repository. The script helps you to install the network policies for your CP4BA production deployment. For more information about downloading `cert-kubernetes`, see [Preparing for a production deployment](#).

The `cert-kubernetes/scripts/cp4a-network-policies.sh` script has three modes for a production deployment type:

#### generate

The generate mode copies the network policy files from the operators to the local subfolder named "network-policies".

**Note:** Network policy files are generated only for the capabilities that are included in your deployment. Some operators do not have network policy templates.

#### install

The install mode installs the network policies on your cluster.

#### delete

The delete mode removes the network policies from your cluster.

The script can be run with the following options:

```
sh ./cp4a-network-policies.sh --help
Usage:
cp4a-network-policies.sh -m [modeType] -n [cp4ba_namespace]
Options:
-h Display help
-m Required: The valid mode types are: [generate], [install], [delete]
-n Required: The target namespace of the CP4BA deployment.
    If CP4BA is deployed using separate namespaces for operators and operands/
    services, the value is the namespace where CP4BA operands/services are deployed.
Additional Information:
```

STEP 1: Run the script in [generate] mode. This copies the sample network policy templates to folder [cert-kubernetes/scripts/network-policies/<namespace>/templates]  
 STEP 2: Review and modify (if needed) the network policy templates based on your cluster environment.  
 STEP 3: Apply the network policies in you cluster manually or optionally run the script in [install] mode to apply templates in the path [cert-kubernetes/scripts/network-policies/<namespace>/templates]

Run the script in generate mode to create files in the following structure.

**Tip:** The folder names match the component names that are defined in the CP4BA CR.

```
network-policies
├── <cp4ba-namespace>
│   ├── logs
│   └── templates
│       ├── Content
│       │   ├── both
│       │   ├── egress
│       │   └── ingress
│       ├── CP4BA
│       │   ├── egress
│       │   └── ingress
│       └── .....
└── .....
```

Use the following steps to install network policies for your CP4BA deployment on your cluster.

## Procedure

1. Log in to the cluster as the <cluster-admin> user.

Using the Red Hat OpenShift CLI, run the following command.

```
oc login https://<cluster-ip>:<port> -u <cluster-admin> -p <password>
```

On ROKS, if you are not already logged in, run the following command.

```
oc login --token=<token> --server=https://<cluster-ip>:<port>
```

2. Make sure that you are in the scripts folder under cert-kubernetes on the client machine you used to connect to the cluster.

```
cd $PATH_TO_EXTRACTED_FILES/cert-kubernetes/scripts
```

3. Create an environment variable to locate the target CP4BA namespace (cp4ba-project).

```
export NAMESPACE=<cp4ba-project>
```

**Note:** If you chose to separate duties of the CP4BA operators and operands, use the project that you set for the \$CP4A\_OPERAND\_NAMESPACE.

4. Run the cp4a-network-policies.sh script in generate mode to copy the network policies from the operators to the network-policies subfolder on the local machine.

```
./cp4a-network-policies.sh -m generate -n $NAMESPACE
```

5. Use the comments in the sample files to review, modify (if necessary), and complete necessary information. Enter values in all the fields that contain the placeholder value of <Required>.

Run the following command to find all the occurrences of <Required>.

```
grep -ir "required" <script_path>/network-policies/$NAMESPACE/templates
```

Where <script\_path> is the path to the cp4a-network-policies.sh file.

The following example shows the comments that are generated in the `egress/templates/dpe/` folder.

```
# This network policy is for Document Processing backend to communicate to the database(s).
# This network policy is specific to the dc_ca_datasource section of the CR.
# You may adjust the egress' section to fit your requirement. For example, if your database
# is container-base and deployed in the same cluster but in a different ns, you may want to
# try to use
# egress:
#   - to:
#     - podSelector: {}
#       namespaceSelector:
#         matchExpressions:
#           - key: kubernetes.io/metadata.name
#             operator: In
#             values:
#               - "<DB namespace>"
# Below is the sample network policy that based on cidr range.
# DO NOT modify the podSelector as it may affect the communication.
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: "$CP4A_OPERAND_NAMESPACE"
  name: '$META_NAME-aca-netpol-db'
  labels:
    app.kubernetes.io/name: "$META_NAME-aca"
    app.kubernetes.io/component: "ACA"
    app.kubernetes.io/instance: "$META_NAME-aca"
    app.kubernetes.io/version: "25.0.0"
spec:
  podSelector:
    matchExpressions:
      - key: app.kubernetes.io/name
        operator: In
        values:
          - '$META_NAME-aca'
          - 'ibm-dpe-operator'
  egress:
    - to:
      - ipBlock:
          cidr: "" #<Required> Fill out the database cidr range such as 1.2.3.4/32
      ports:
        - protocol: TCP
          port: #<Required> Fill out the port number for database

  policyTypes:
    - Egress
```

6. Run the script in the "install" mode.

```
./cp4a-network-policies.sh -m install -n $NAMESPACE
```

**Note:** If your IBM Cloud Pak for Business Automation deployment includes IBM Business Automation Workflow, restart the pods to maintain successful connectivity with the Business Automation Workflow database.

7. Install network policies for Cloud Pak foundational services. For more information, see [Installing network policies for foundational services](#).

If a network policy restriction is in place, the Cloud Pak foundational services might have connectivity issues.

## What to do next

If you need to delete the network policies for any reason, run the `cp4a-network-policies.sh` script in delete mode.

```
./cp4a-network-policies.sh -m delete -n $NAMESPACE
```

The script deletes only the network policies that are created by the CP4BA operators and found in the `cert-kubernetes/scripts/network-policies/<namespace>/templates` folder. Any other network policies that you created are not deleted.

To uninstall the network policies of the Cloud Pak foundational services, see [Uninstalling network policies for foundational services](#).



**Attention:** If you plan to modify the CR after you installed the network policies, the update might need new network policies. Therefore, re-run the `cp4a-network-policies.sh` script with the `generate` and `install` modes to apply the new network policies. Always wait for the reconciliation to complete successfully before you run the script again. Confirm that all the components in the CR are successfully installed and can be retrieved by checking the status section of the applied CR file.

## Validating your production deployment

Instead of manually checking all the URLs and certificates that are created by your deployment, you can run a script to validate these objects automatically in a few minutes.

### Before you begin

Make sure that your client machine can connect to the cluster you want to use, and has the necessary tools. Install the appropriate tools from the following list.

#### jq (a JSON processor open source tool)

- On macOS:

1. How to install jq.

```
brew install jq
```

2. Verify the installation.

```
jq --version  
jq --help
```

- On CentOS/RHEL:

1. Install the EPEL Repository.

```
sudo yum install epel-release
```

2. Update the packages.

```
sudo yum update
```

3. Install jq.

```
sudo yum install jq
```

4. Verify the installation.

```
jq --version  
jq --help
```

### About this task

The post-installation script (`cp4a-post-install.sh`) is found in the `cert-kubernetes` repository. The script helps you to assess the readiness of your CP4BA deployment, and to retrieve and validate connection information to all its services. For more information about downloading `cert-kubernetes`, see [Preparing for a production deployment](#).

The `cert-kubernetes/scripts/cp4a-post-install.sh` script has four modes for a production deployment type:

#### precheck

The `precheck` mode gets some basic information about the cluster, the console, and the client.

## status

The status mode gets the status of all the components from the custom resource (Ready | Not Ready | Not Installed).

## console

The console mode gets the console connection information from the URLs and credentials.

## probe

The probe mode checks the readiness and health of the deployment endpoints.

The script can be run with the following options:

```
./cp4a-post-install.sh --help
--precheck      This mode gives information about the cluster and the client.
--status        This mode gives the status of the services of the deployment.
--console       This mode gives the service URLs of the consoles in the deployment.
--probe         This mode checks the readiness of the deployment endpoints.
```

When you run the script in the status, console, or probe mode, the commands display information about the Cloud Pak for Business Automation version and interim fix number of the installed deployment. The output also includes the list of CP4BA capabilities that are installed.

If no CP4BA production deployments are found on the cluster, then information about the cluster is displayed along with the following message:

```
No resources found for CP4BA Production deployment types.
```

If no CP4BA production deployments are found in the namespace in which you run the script, then information about the cluster is displayed along with the following message:

```
No CP4BA Production deployment found in namespace NAMESPACE.
```

## Procedure

1. Make sure that you downloaded the cert-kubernetes repository to a Linux based machine (CentOS Stream/RHEL/MacOS) or a client to a Linux-based machine.
2. Make sure that you are in the scripts folder under cert-kubernetes.
3. Log in to the target cluster as the <cluster-admin> user.

Using the Red Hat OpenShift CLI:

```
oc login https://<cluster-ip>:<port> -u <cluster-admin> -p <password>
```

On ROKS, if you are not already logged in:

```
oc login --token=<token> --server=https://<cluster-ip>:<port>
```

4. Check the values for the environment variables in the scripts/helper/post-install/env.sh script under cert-kubernetes.

The variable that is the most important to check is for the foundational services namespace, as it can be customized and can be either cluster-scoped or namespace-scoped. The default value is set to `ibm-common-services`, which is for a cluster-scoped instance. If you installed foundational services in a namespace-scoped instance, you must change the value for your CP4BA deployment. The following example sets `cp4ba-project` as the namespace-scoped instance.

```
CP4BA_COMMON_SERVICES_NAMESPACE=cp4ba-project
```

5. Run the script in the "precheck" mode.

```
./cp4a-post-install.sh --precheck
```

6. Run the script in the "status" mode.

```
./cp4a-post-install.sh --status
```

The status of the listed components can be one of the following values:

- Ready is used to indicate that the component is installed successfully and ready to use.
- Not Ready is used to indicate that the component is not installed yet and is not ready to be used.
- Not Installed is used to indicate that the component is not included in the CP4BA deployment.
- Not found is used when the status value of the resource is missing. A Not found value usually indicates that the operator is changing the status.

7. Run the script in the "console" mode.

```
./cp4a-post-install.sh --console
```

The output provides valuable information about the available consoles, which can be shared and distributed to administrators and users who request access to the Cloud Pak for Business Automation capabilities.

8. Run the script in the "probe" mode to know whether you can access and send requests to the deployment endpoints.

- a. Add your values to the following parameters in the `env.sh` file, which is located under the `scripts/helper/post-install` folder.

**PROBE\_USER\_API\_KEY**

A Platform API key that is generated for a specified user. For more information, see [Generating API keys for authentication](#).

**PROBE\_USER\_NAME**

The name of the user in the API key. This user must have access rights to all the Cloud Pak for Business Automation applications.

**PROBE\_USER\_PASSWORD**

The basic authentication password of the user.

**PROBE\_VERBOSE**

Can be empty or enabled with `'-v'` to include extra debugging information. Extra log information can be useful to determine why an endpoint is not ready to receive traffic.

- b. Then, run the following command:

```
./cp4a-post-install.sh --probe
```

**Note:** Keep in mind that when you run the probe, OpenShift sends traffic to the pods only if the probe succeeds.

The probe mode returns one of the following messages for each endpoint:

**OK**

The probe succeeded. The endpoint is accessible by using a `cURL` command and the name of the application is returned.

**BAD: <tested URL>**

The probe failed. The endpoint is not accessible by using a `cURL` command. You can run the command from a command terminal to check that the endpoint is still inaccessible.

**Unauthorized: <tested URL>**

The probe failed due to user authorization. You can run the script again to check that the credentials are consistently rejected.

**Not installed**

Access to the endpoint failed. The cause of the failure can be `Not Found`, `Forbidden`, or `Service Unavailable`.

**Cannot verify**

The `cURL` command cannot be used to verify the endpoint. You must use a web browser to validate the endpoint.

## What to do next



**Attention:** If you see connection errors to the external services, such as databases or an LDAP, after you installed your CP4BA deployment, then create an egress network policy to allow communication between your CP4BA deployment and the external services. The connection errors occur only when the external services are on the same cluster as your CP4BA deployment and in different namespaces. For more information, see [Network policies to manage access to external services \(egress\)](#).

## Completing post-installation tasks

For some deployments, extra steps are needed to make sure that the environment works correctly.

### About this task



**Warning:** Whenever you modify your CP4BA multi-pattern deployment, always use the top-level custom resource (CR) YAML file (ICP4ACluster). All the relevant changes are passed down to the lower-level CRs. If you update, for example, the FNCM CR YAML file (Content) directly in your CP4BA multi-pattern deployment, then these changes are overwritten when the ICP4ACluster CR is reconciled. To update data source sections for the FNCM capability in a CP4BA multi-pattern deployment, edit the ICP4ACluster CR and the CP4BA operator then passes these changes to the lower-level operator.

Always use the CP4BA multi-pattern CR to add or remove capabilities in a CP4BA multi-pattern deployment. You cannot install a CP4BA FNCM deployment and add capabilities, like ADS or ADP, to it.

The following progress bar shows you where you are in the installation process.



## Completing post-installation tasks for Workflow Process Service Authoring

You must complete post-installation tasks before you can run IBM Workflow Process Service Authoring.

### Procedure

1. Complete the steps in [“Completing post-installation tasks for Business Automation Studio”](#) on page 32.
2. Verify your installation and then log in to the web applications to get started.
  - a. Wait for the installation to complete. The installation might take around 45 minutes, depending on your computer's performance. Then, check the status of the pods. Get the names of the pods that were deployed by running the following command:

```
oc get pod -n <project_name>
```

The following is an example of a successful pod status deployed with a small profile. The pods are all running or completed, depending on the type of pod.



Name	READY	STATUS	RESTARTS	AGE
cloud-native-postgresql-catalog-76jzt	1/1	Running	0	44h
common-service-db-1	1/1	Running	0	43h
common-service-db-2	1/1	Running	0	43h
common-web-ui-74bfc5df85-hwqsv	1/1	Running	0	43h
create-postgres-license-config-4hfrw	0/1	Completed	0	43h
create-secrets-job-jmdxp	0/1	Completed	0	43h
flink-kubernetes-operator-657cd8d449-6nsdv	2/2	Running	0	43h
iaf-system-entity-operator-9d559d89-xstsk	2/2	Running	0	43h
iaf-system-kafka-0	1/1	Running	0	43h
iaf-system-zookeeper-0	1/1	Running	0	43h
iam-config-job-fwcvb	0/1	Completed	0	43h
ibm-ads-operator-6bf55dcdc-fqz4f	1/1	Running	0	44h
ibm-bts-cnpg-a1-cp4ba-bts-1	1/1	Running	0	43h
ibm-bts-cnpg-a1-cp4ba-bts-2	1/1	Running	0	43h
ibm-bts-cp4ba-bts-316-deployment-77dc794477-4kzh4	1/1	Running	0	43h
ibm-bts-cp4ba-bts-316-deployment-77dc794477-x8rhz	1/1	Running	0	43h
ibm-bts-operator-catalog-v3-35-znpbj	1/1	Running	0	44h
ibm-bts-operator-controller-manager-6b787bf56-8ncrj	1/1	Running	0	43h
ibm-common-service-operator-6864ddd555-7bhql	1/1	Running	0	44h
ibm-commonui-operator-684c8b9986-jcbz9	1/1	Running	0	43h
ibm-content-operator-786d86cc4d-2kcd6	1/1	Running	0	44h
ibm-cp4a-operator-59c684f787-j4vtb	1/1	Running	0	40h
ibm-cp4a-operator-catalog-cpvv2	1/1	Running	0	44h
ibm-cp4a-wfps-operator-7b7d8df58-mz9cr	1/1	Running	0	44h
ibm-cs-install-catalog-v4-12-0-szwvg	1/1	Running	0	44h
ibm-dpe-operator-f99694c98-dndzv	1/1	Running	0	44h
ibm-events-operator-catalog-v5-1-0-xkzs4	1/1	Running	0	44h
ibm-events-operator-v5.1.2-5f64978fcb-p866f	1/1	Running	0	43h
ibm-fncm-operator-catalog-948lx	1/1	Running	0	44h
ibm-iam-operator-5898599478-st4wm	1/1	Running	0	43h
ibm-iam-operator-catalog-4-11-0-59lvj	1/1	Running	0	44h
ibm-insights-engine-operator-599f5b4564-rz4b7	1/1	Running	0	44h
ibm-nginx-c6445696d-lmfzw	2/2	Running	0	43h
ibm-nginx-tester-ccc754858-6kgxp	2/2	Running	0	43h
ibm-odm-operator-5c795cb9b8-pvb2n	1/1	Running	0	44h
ibm-opencontent-flink-mnk4d	1/1	Running	0	44h
ibm-opensearch-operator-catalog-zfwwt	1/1	Running	0	44h
ibm-opensearch-operator-controller-manager-65cdd9cbb6-kbhj	1/1	Running	0	43h
ibm-pfs-operator-5d47d647fc-vhrq4	1/1	Running	0	44h
ibm-workflow-operator-7d56c8d58c-pp47c	1/1	Running	0	44h
ibm-zen-operator-8dbb57fb4-nb2pj	1/1	Running	0	43h
ibm-zen-operator-catalog-6-1-0-ffz59	1/1	Running	0	44h
icp4a-foundation-operator-66df74c998-crr6z	1/1	Running	0	44h
icp4adeploy-bai-bpmn-ghdl6	0/1	Completed	0	43h
icp4adeploy-bai-setup-rpprj	0/1	Completed	0	43h
icp4adeploy-bastudio-bootstrap-dk7hk	0/1	Completed	0	43h
icp4adeploy-bastudio-deployment-0	1/1	Running	0	39h
icp4adeploy-bastudio-ltpa-d4ctw	0/1	Completed	0	43h
icp4adeploy-bastudio-zen-translation-p5wm7	0/1	Completed	0	43h
icp4adeploy-insights-engine-application-setup-tilfw	0/1	Completed	0	43h
icp4adeploy-insights-engine-cockpit-6fcb6847df-2cndz	1/1	Running	1 (27h ago)	43h
icp4adeploy-insights-engine-flink-87d465bbc-klrbz	2/2	Running	0	43h
icp4adeploy-insights-engine-flink-taskmanager-7d7767f567-jkbrk	1/1	Running	0	43h
icp4adeploy-insights-engine-management-857b95f6dd-4nnzf	2/2	Running	0	43h
meta-api-deploy-6fc85758c5-4qnvw	1/1	Running	0	43h
oidc-client-registration-gqv9t	0/1	Completed	0	43h
opensearch-all-000	1/1	Running	0	43h
opensearch-all-001	1/1	Running	0	43h
opensearch-all-002	1/1	Running	0	43h
opensearch-snapshot-repo-vkdn1	0/1	Completed	0	11m
operand-deployment-lifecycle-manager-6d87fd57-fspjs	1/1	Running	0	43h
platform-auth-service-7b9fcc7dbb-dtgtx	1/1	Running	0	43h
platform-identity-management-89db6755c-6j9ml	1/1	Running	0	43h
platform-identity-provider-575fff4f7c-lsssh	1/1	Running	0	43h
postgresql-operator-controller-manager-1-25-1-76cd94d477-5gbl1	1/1	Running	0	43h
setup-job-h68sz	0/1	Completed	0	43h
usermgmt-5f4767788f-hlxs6	1/1	Running	0	43h
usermgmt-ensure-tables-job-2x8l5	0/1	Completed	0	43h
zen-audit-7b5d599d48-dq7q2	1/1	Running	0	43h
zen-core-688c66845c-g6dt9	2/2	Running	0	43h
zen-core-api-59b697f7df-29r5k	2/2	Running	0	43h
zen-core-create-tables-job-rl9sx	0/1	Completed	0	43h
zen-core-pre-requisite-job-xzdk9	0/1	Completed	0	43h
zen-metastore-backup-cron-job-29159040-zrmrp	0/1	Completed	0	23h
zen-metastore-backup-cron-job-29159520-q7dhw	0/1	Completed	0	15h
zen-metastore-backup-cron-job-29160000-vvvpv	0/1	Completed	0	7h50m
zen-metastore-edb-1	1/1	Running	0	43h
zen-metastore-weekly-backup-cron-job-29158620-5nmh7	0/1	Completed	0	30h
zen-minio-0	1/1	Running	0	43h
zen-minio-1	1/1	Running	0	43h
zen-minio-2	1/1	Running	0	43h
zen-minio-create-buckets-job-chsm9	0/1	Completed	0	43h
zen-pre-requisite-job-dzsbr	0/1	Completed	0	43h
zen-remote-svc-inst-status-cron-job-29160456-9h8sd	0/1	Completed	0	14m
zen-remote-svc-inst-status-cron-job-29160462-d9pd2	0/1	Completed	0	8m54s
zen-remote-svc-inst-status-cron-job-29160468-9r72r	0/1	Completed	0	2m54s
zen-route-custom-cert-refresh-cronjob-29159040-v99b7	0/1	Completed	0	23h
zen-route-custom-cert-refresh-cronjob-29159520-cqg6n	0/1	Completed	0	15h
zen-route-custom-cert-refresh-cronjob-29160000-d4xh7	0/1	Completed	0	7h50m
zen-svc-inst-status-cron-job-29160460-6429k	0/1	Completed	0	10m
zen-svc-inst-status-cron-job-29160464-28lzg	0/1	Completed	0	6m54s
zen-svc-inst-status-cron-job-29160468-fk8wh	0/1	Completed	0	2m54s
zen-watcher-55db4fdd68-p6ng6	2/2	Running	0	43h

- b. Log in to Business Automation Studio. To get the URL and the user credentials to log in, access the Config Maps from the OpenShift web console.

i) Log in to your cluster web console.

- ii) Select your namespace.
  - iii) In the left panel, select **Workloads > Config Maps**.
  - iv) Find cp4ba-access-info. You can get IBM Cloud Pak dashboard information from bastudio-access-info.
  - v) Access Business Automation Studio by using the URL in bastudio-access-info.
3. To customize Workflow Process Service Authoring, see [Customizing Business Automation Studio properties](#).
  4. If you already installed Workflow Process Service Runtime, you can configure it to work with Workflow Process Service Authoring.
  5. Optional: If you want to include the data collector and data indexer, add pfs to the `spec.shared_configuration.sc_optional_components` section of your custom resource.
  6. Optional: If you want to use an external Elasticsearch, configure it under the `workflow_authoring_configuration` section of your custom resource. For more information about the Elasticsearch parameters, see [External Elasticsearch configuration parameters](#).

## What to do next

If Workflow Process Service Authoring is in its own namespace, you can see the authoring server's tasks in the embedded Workplace.

If you deployed the application pattern together with Workflow Process Service Authoring, you can also see the authoring server's tasks in the Workplace in IBM Business Automation Navigator.

## Completing post-installation tasks for Cloud Pak for Business Automation foundation

---

All capabilities contain one or more components from the Cloud Pak foundation pattern. You must check to see which components are included with your capability and complete the post-installation tasks.

### Completing post-installation tasks for Business Automation Studio

Post installation you can customize the IBM Business Automation Studio Liberty properties or modify the `100Custom.xml` configuration properties. If you prepared the Application Designer configuration with Business Automation Workflow before the installation, you can now complete the configuration to enable Business Automation Workflow to communicate with Business Automation Studio.

## Changing default values for Cloud Pak foundational services

---

Each instance of Cloud Pak for Business Automation that you install includes Cloud Pak foundational services, or locates an already installed instance. The foundational services can be configured post-installation to better integrate with Cloud Pak for Business Automation.

### About this task

The foundational services help you manage and administer IBM software on your cluster. For example, the Cloud Pak foundational services include services such as the Platform UI (Zen) Service, the License Service, and the Identity Management (IM) Service. From the Platform UI (Zen), you can view key metrics for components of IBM Cloud® Paks and Cloud Pak foundational services that are installed on the cluster.

You might want to configure the foundational services in the following ways.

#### Cluster administrator password

The cluster administrator password is stored in a Kubernetes secret. You can change the auto-generated password and restart the services that use the password by running a `cloudctl` command.

**Note:** IBM Cloud Pak® CLI (cloudctl) is useful to view information about your cluster and to manage your cluster. To install cloudctl, see [Installing IBM Cloud Pak® CLI \(cloudctl\)](#).

```
cloudctl pm update-secret kube-system platform-auth-idp-credentials -d admin_password
```

The password must follow the defined password rules. To list the password rules, run the following command:

```
cloudctl pm password-rules <namespace>
```

## Platform UI (Zen) customization

The Zen Service is a reverse proxy that provides a common external URL to access Cloud Pak for Business Automation capabilities. For more information, see [Platform UI](#). Zen uses specific roles to define who can access a particular interface. For more information about managing the roles and user permissions, see [Managing users](#).

**Note:** Each Cloud Pak has its own documentation on the roles that apply to its interfaces. For some APIs and UIs, you might need extra authorization in addition to the Platform UI roles. Therefore, it is best to review the documentation of a particular API or UI before you use it.

The `ibm-zen-operator` manages the Zen Service, and the Cloud Pak for Business Automation operator does not manage the Zen settings after the initial creation. Customization of the Zen Service must be made directly in the Zen custom resource (CR).

You can review the settings and health of the Zen Service by running the following command, where `<namespace>` is usually the Cloud Pak for Business Automation namespace.

```
oc get zenservice -o yaml -n <namespace>
```

The route that is associated with Zen is called `cpd` and it uses secure TLS communications. If you need the root CA for an external truststore, see [Exporting the Zen CA and common services CA](#). When you access the `cpd` route, login requests are redirected to the IM service and the `cp-console` route. You can customize the hostname and certificates for these routes. For more information, see [Customizing the Cloud Pak Identity Management \(IM\) service](#).

All the TLS certificates for Cloud Pak foundational services are created during installation, but you can replace the certificate and the route hostname for the foundational services entry point. The entry point is the endpoint that is used to access the console from outside the cluster. For more information, see [Customizing the Cloud Pak entry point](#).

## License Service custom certificates

To avoid certificate issues and untrusted certificates when you access the UIs and REST APIs, you can configure a custom certificate for License Service communication.

1. Change the certificate name to `tls.crt`.
2. Change the name of the key to `tls.key`.
3. Run the following command to change the directory to where the certificate and the key are stored:

```
cd <certificate_directory>
```

4. Create a secret by using the following command:

```
licensingNamespace=$(oc get pods --all-namespaces | grep "ibm-licensing-service-" | awk  
{'print $1'})  
oc create secret tls ibm-licensing-certs --key tls.key --cert tls.crt -n $  
{licensingNamespace}
```

5. Edit the IBM Licensing custom resource to enable the https connection.

```
apiVersion: operator.ibm.com/v1alpha1  
kind: IBMLicensing  
metadata:  
  name: instance
```

```
spec:
  httpsEnable: true
```

To access the IBM Licensing operator in the OpenShift console:

- a. From the navigation menu, click **Operators > Installed Operators > IBM Licensing Operator**.
  - b. Click the **IBM License Service** tab.
  - c. Click the **IBMLicensing** instance, and then click the **YAML** tab.
6. To apply the custom certificate that you created, add the **httpsCertsSource** parameter:

```
apiVersion: operator.ibm.com/v1alpha1
kind: IBMLicensing
metadata:
  name: instance
spec:
  httpsEnable: true
  httpsCertsSource: ibm-licensing-certs
```

### Mutual authentication for LDAP connections

Mutual Transport Layer Security (mTLS) requires both the client and the server to present their own certificates to each other and verify the identity of each other before a secure connection is opened. For the LDAP mutual TLS authentication to work, you must allow your LDAP server to import the CP4BA signer certificate otherwise the session is immediately cancelled. For more information, see [Configuring mutual TLS authentication between IM and an LDAP server](#).

### Default storage class

The storage class that you used for the ICP4ACluster instance is used for the PostgreSQL database (common-service-db). You can change the storage class name by adding the **storageClass** parameter with a storage class of your choice in the OperandConfig instance in the foundational services namespace.

**Note:** If you used a namespace-scoped foundational services instance, then the namespace is defined in the common-service-maps configMap for the CP4BA deployment. If you used a cluster-scoped instance, then the namespace is `ibm-common-services`.

You can access the common-service instance by using the OpenShift console or by using the command-line interface (CLI).

- In the console, use the following steps:
  1. From the navigation menu, click **Operators > Installed Operators > Operand Deployment Lifecycle Manager > OperandConfig**.
  2. Click the overflow menu icon of the common-service instance, and click **Edit OperandConfig**.
- To use the CLI, run the following command:

```
oc edit OperandConfig common-service -n <namespace>
```

## Completing extra post-installation tasks on ROKS

For deployments on Red Hat OpenShift Kubernetes Service (ROKS), routes are automatically generated. If you want to change the CA certificate, then extra steps are needed to ensure that the environment works correctly.

### About this task

The access information ConfigMap includes the list of routes that can be used by external sources to access the deployed services. For information about the routes and the services they access, including URL path prefixes to use with the Zen front door, see the capability post-installation tasks.

# Optional: Installing IBM Workflow Process Service Runtime production deployments

Workflow Process Service Runtime is a small-footprint business automation environment for testing and running workflow processes that coordinate manual tasks and services.

You can install Workflow Process Service Runtime on Red Hat OpenShift Container Platform. The Workflow Process Service Runtime operator catalog in the OpenShift Container Platform OperatorHub provides a user interface for you to install a production deployment with an operator lifecycle manager (OLM). Workflow Process Service Runtime includes a preconfigured PostgreSQL database. For identity and access management, you can choose to use LDAP or you can choose a third-party provider. The third-party providers include System for Cross-domain Identity Management (SCIM) support.

To install Workflow Process Service Runtime on Kubernetes, complete the following steps. The steps include the process to prepare, deploy, and configure a Workflow Process Service Runtime server.

If you run into issues while installing Workflow Process Service Runtime, see [Troubleshooting Workflow Process Service](#).

If you are upgrading, see [Upgrading Workflow Process Service Runtime deployments](#).

## Deploying required Workflow Process Service Runtime components

To install Workflow Process Service Runtime, you must use the Cloud Pak for Business Automation operator to configure Resource Registry and root Certificate Authority (CA).

If you already installed one of the Cloud Pak for Business Automation deployment patterns (following the steps in [Creating a production deployment](#)), validate your deployment to make sure it includes the following configurations.

1. If your cluster does not support dynamic provisioning, create your persistent volume (PV) manually for your IBM Resource Registry component by completing the steps in [Optional: Implementing storage](#).

If your cluster has dynamic storage provisioning that is enabled, create the following .yaml file, and fill in the values of `sc_slow_file_storage_classname`, `sc_medium_file_storage_classname`, and `sc_fast_file_storage_classname`.

```
apiVersion: icp4a.ibm.com/v1
kind: ICP4ACluster
metadata:
  name: icp4adeploy
  labels:
    app.kubernetes.io/instance: ibm-dba
    app.kubernetes.io/managed-by: ibm-dba
    app.kubernetes.io/name: ibm-dba
    release: 25.0.0
spec:
  appVersion: 25.0.0
  ibm_license: "accept"
  shared_configuration:
    ## Use this parameter to specify the license for the IBM Cloud Pak for Business
    ## Automation deployment for the rest of the Cloud Pak for Business Automation components.
    ## This value could differ from the rest of the licenses.
    sc_deployment_license: production
    sc_deployment_type: production
    ## The user script will populate these three (3) parameters based on your input for
    ## "production" deployment.
    ## If you manually deploying without using the user script, then you need to provide
    ## the different storage classes for the slow, medium, and fast storage parameters below.
    ## If you only have 1 storage class defined, then you can use that storage class for
    ## all 3 parameters.
    ## sc_block_storage_classname is for Zen, Zen requires/recommends block storage (RW0)
    ## for metastoreDB
    storage_configuration:
      sc_slow_file_storage_classname: "<Required>"
      sc_medium_file_storage_classname: "<Required>"
      sc_fast_file_storage_classname: "<Required>"
```

```

    sc_block_storage_classname: "<Required>"
    sc_deployment_platform: OCP
    ## this field is required to deploy Resource Registry (RR)
    resource_registry_configuration:
        replica_size: 1

```

2. If you want to configure one or more LDAP configurations, use the `ldap_configuration` parameter in the `icp4acluster` CR. For more information about LDAP configuration, see [LDAP configuration](#). You can also configure LDAP in post-deployment by using the Common UI console.
3. If you made changes, wait a few minutes, then run the command `oc get icp4acluster -o yaml` to make sure that the root certificate authority and Resource Registry are ready. Make sure that `.status.components.prereq.rootCAStatus` is `Ready` and `.status.components.prereq.rootCASecretName` is filled with the correct secret name. Make sure that `.status.endpoints["Resource Registry"]` appears in the endpoints list. For example:

```

status:
  components:
    ...
    prereq:
      conditions: []
      rootCASecretName: icp4adeploy-root-ca
      rootCAStatus: Ready
    resource-registry:
      rrAdminSecret: icp4adeploy-rr-admin-secret
      rrCluster: Ready
      rrService: Ready
    ...
  endpoints:
  - name: Resource Registry
    scope: Internal
    type: gRPC
    uri: icp4adeploy-dba-rr-client:2379

```

4. Make sure that Zen and Resource Registry pods are listed in the `oc get pod` command result. For example:

NAME	READY	STATUS	
RESTARTS	AGE		
01b20e7d07a23fdc5fc5daffd3d3a01b67ae9f2f3f0177c5aa7e99a1b176x510	0/1	Completed	9d
10f7349b6d490b712d6e54b22c31ce892381d204f7418a1f46442a58f92fdqh0	0/1	Completed	9d
1f49b1adb4ec963b0360c79237f45ab7c9b2ceb6d9085d888f6da0e1d1vf7rw0	0/1	Completed	9d
233900532d458faae6f637b54f920859c860cd3677394fd06d4753b0976f91s0	0/1	Completed	9d
26ec29312051b5bdbc8d356c36508f9f8d45e448388e0cf4e2d53e310dpnknr0	0/1	Completed	9d
32ce47ba45b048f042cf1d5f5cf678cc3436a2a76a1023a88e3da63e49r96xc0	0/1	Completed	9d
54832581e5a562f16b2b34bcbd55fc3c76189c80c4a7ec41a1be9f8279ctpmn0	0/1	Completed	9d
6a45818f06cf7d9e6b04574bf439f0e95d448300f396c9795a8dff3fa91tlhf0	0/1	Completed	9d
7982bb49dbfe94248a8328ff9dfbbe84e5bc527764c17e3f0704e5ecccqvwh60	0/1	Completed	9d
7cef7438bdb68f037d72351e5494ea2749af976ca9c33bcb0b1df94bc96smk80	0/1	Completed	9d
878dafdf528656ab8f8f43d0be4e116b94bec9201ff582575a0a4e3ba8lbrhs0	0/1	Completed	9d
90bb458296ae89a3200173317fb6a219ac899cd35a9f1a0d3a65bfbf2a9gs4m0	0/1	Completed	9d
96bf47f180216abbeeb3a0b4757ef6abda50d874c9f439419140e056efvssct0	0/1	Completed	9d
b047dc22fdfaaf0528c2f92c4fd2ea6b9684a5e2a1a10ce3f588d262abc4knf0	0/1	Completed	9d
b19be3f0da08098d4149646967f05fe753260ae9bc60796f76ba54239c97ncm0	0/1	Completed	9d
c904e2eec7c1d7c8156b1dc44edd36345b27eaae25fde1124e4780d885qqnj50	0/1	Completed	9d
cloud-native-postgresql-catalog-8mz7g	1/1	Running	9d
common-service-db-1	1/1	Running	9d
common-service-db-2	1/1	Running	9d

(2d1h ago) 9d common-web-ui-54dfd5bc4c-4fkxh 0 9d	1/1	Running	
create-postgres-license-config-jfprc 0 9d	0/1	Completed	
create-secrets-job-9rv2h 0 9d	0/1	Completed	
d293451c834b2457665945595765021331ebfb61f8c78373b0d7e4646446jss 0 9d	0/1	Completed	
d4e56d959a1c411da8e52fce1f0a81058dcf63dd96eaf13cc3bf9053bfjxb9x 0 9d	0/1	Completed	
e640ffca4b8e31cbca32d081786200a9f56f7649d051bde8939531d299tr6nc 0 9d	0/1	Completed	
fa3692f7e51b2788a1825b587f34716dc555317af6908c80a40faff9a4mpsmn 0 9d	0/1	Completed	
flink-kubernetes-operator-769bb559cb-dfz9l 0 9d	2/2	Running	
iaf-system-entity-operator-6979d5ffb-lnxt2 0 9d	2/2	Running	
iaf-system-kafka-0 0 9d	1/1	Running	
iaf-system-zookeeper-0 0 9d	1/1	Running	
iam-config-job-66cnh 0 9d	0/1	Completed	
ibm-ads-operator-865d866457-kct7v ago) 9d	1/1	Running	2 (22h)
ibm-bts-cnpg-a1-cp4ba-bts-1 (2d15h ago) 9d	1/1	Running	1
ibm-bts-cnpg-a1-cp4ba-bts-2 (2d1h ago) 9d	1/1	Running	1
ibm-bts-cp4ba-bts-316-deployment-7d58bdccb6-xdbf8 0 9d	1/1	Running	
ibm-bts-cp4ba-bts-316-deployment-7d58bdccb6-xqlqg (7d22h ago) 9d	1/1	Running	1
ibm-bts-operator-catalog-v3-35-78rbs 0 9d	1/1	Running	
ibm-bts-operator-controller-manager-55cfbc78c7-7h276 ago) 9d	1/1	Running	2 (22h)
ibm-common-service-operator-7b85f4f7c5-c975s 0 9d	1/1	Running	
ibm-commonui-operator-684c8b9986-qqgw4 0 9d	1/1	Running	
ibm-content-operator-68774b947-kmhcs 0 9d	1/1	Running	
ibm-cp4a-operator-5d686d5cc6-dh9qf 0 9d	1/1	Running	
ibm-cp4a-operator-catalog-74t84 0 9d	1/1	Running	
ibm-cp4a-wfips-operator-56cc557d5c-5bpqg 0 9d	1/1	Running	
ibm-cs-install-catalog-v4-12-0-fx8wh 0 9d	1/1	Running	
ibm-dpe-operator-8679658b8c-hgcpq 0 9d	1/1	Running	
ibm-events-operator-catalog-v5-1-0-m42nm 0 9d	1/1	Running	
ibm-events-operator-v5.1.2-5f64978fcb-shtvw 0 9d	1/1	Running	
ibm-fncm-operator-catalog-5k2fx 0 9d	1/1	Running	
ibm-iam-operator-5898599478-cjvzx 0 9d	1/1	Running	
ibm-iam-operator-catalog-4-11-0-5b778 0 9d	1/1	Running	
ibm-insights-engine-operator-7f4f96cdc8-zvlmd 0 9d	1/1	Running	
ibm-nginx-7d7cf57b64-fxnbfb 0 9d	2/2	Running	
ibm-nginx-tester-748477db7d-vxkb4 0 9d	2/2	Running	
ibm-odm-operator-79fc7754f9-xmplh ago) 9d	1/1	Running	1 (22h)
ibm-opencontent-flink-j65gp 0 9d	1/1	Running	
ibm-opensearch-operator-catalog-sjlzf 0 9d	1/1	Running	
ibm-opensearch-operator-controller-manager-5f455f5966-5kc4c ago) 9d	1/1	Running	1 (22h)
ibm-pfs-operator-b4fc4d7c9-5z8vf ago) 9d	1/1	Running	1 (22h)
ibm-workflow-operator-794bcbfbd9-569qb	1/1	Running	1 (22h)



ago) 9d			
ibm-zen-operator-8dbb57fb4-gpkw9	1/1	Running	
0 9d			
ibm-zen-operator-catalog-6-1-0-wxlq7	1/1	Running	
0 9d			
icp4a-foundation-operator-5b854f6bc5-v9s4z	1/1	Running	
0 9d			
icp4adeploy-bai-bpmn-7hp7f	0/1	Completed	
0 9d			
icp4adeploy-bai-setup-nbjqn	0/1	Completed	
0 9d			
icp4adeploy-bastudio-bootstrap-978c5	0/1	Completed	
0 9d			
icp4adeploy-bastudio-deployment-0	1/1	Running	
0 9d			
icp4adeploy-bastudio-ltpa-8pvv4	0/1	Completed	
0 9d			
icp4adeploy-bastudio-zen-translation-xsvtt	0/1	Completed	
0 9d			
icp4adeploy-insights-engine-application-setup-qc586	0/1	Completed	
0 9d			
icp4adeploy-insights-engine-cockpit-64d445d9b5-k5s1l	1/1	Running	
0 9d			
icp4adeploy-insights-engine-flink-7b48bcd9ff-5hfpn	2/2	Running	
0 9d			
icp4adeploy-insights-engine-flink-taskmanager-59bbf99c8b-w2bmq	1/1	Running	
0 9d			
icp4adeploy-insights-engine-management-6c5f66dcf5-xpqxp	2/2	Running	
0 9d			
meta-api-deploy-6fc85758c5-7t6v9	1/1	Running	
0 9d			
oidc-client-registration-g9mwp	0/1	Completed	
0 9d			
opensearch-all-000	1/1	Running	
0 9d			
opensearch-all-001	1/1	Running	
0 9d			
opensearch-all-002	1/1	Running	
0 9d			
opensearch-snapshot-repo-w9cjz	0/1	Completed	
0 14m			
operand-deployment-lifecycle-manager-6d87fd57-nj987	1/1	Running	
0 9d			
platform-auth-service-5898bdd5d6-mx7wd	1/1	Running	
0 8h			
platform-identity-management-657846f669-f7fj6	1/1	Running	
0 8h			
platform-identity-provider-5bd66f6d89-45vzz	1/1	Running	
0 8h			
postgresql-operator-controller-manager-1-25-1-d4c6fb84d-24rlq	1/1	Running	3 (22h
ago) 9d			
setup-job-f7htt	0/1	Completed	
0 9d			
usermgmt-85f796f7d5-c6glw	1/1	Running	
0 8h			
usermgmt-ensure-tables-job-t9gtq	0/1	Completed	
0 9d			
zen-audit-59c764cd7b-f4n99	1/1	Running	
0 9d			
zen-core-84fd5fd44f-qpmfj	2/2	Running	
0 9d			
zen-core-api-5c7bfb4b89-6z4vg	2/2	Running	1
(7d6h ago) 9d			
zen-core-create-tables-job-j9qh	0/1	Completed	
0 9d			
zen-core-pre-requisite-job-zfx7t	0/1	Completed	
0 9d			
zen-metastore-backup-cron-job-29147520-k9prh	0/1	Completed	
0 18h			
zen-metastore-backup-cron-job-29148000-zg5tt	0/1	Completed	
0 10h			
zen-metastore-backup-cron-job-29148480-dbk42	0/1	Completed	
0 177m			
zen-metastore-edb-1	1/1	Running	3
(2d1h ago) 9d			
zen-metastore-monthly-backup-cron-job-29148600-5j72d	0/1	Completed	
0 57m			
zen-metastore-weekly-backup-cron-job-29138460-xmchh	0/1	Completed	
0 7d1h			
zen-metastore-weekly-backup-cron-job-29148540-djvzd	0/1	Completed	
0 117m			
zen-minio-0	1/1	Running	



```

0          9d
zen-minio-1          1/1    Running
0          9d
zen-minio-2          1/1    Running
0          9d
zen-minio-create-buckets-job-khv4g          0/1    Completed
0          9d
zen-pre-requisite-job-lpfr9          0/1    Completed
0          9d
zen-remote-svc-inst-status-cron-job-29148642-7h2qn          0/1    Completed
0          15m
zen-remote-svc-inst-status-cron-job-29148648-85r8w          0/1    Completed
0          9m41s
zen-remote-svc-inst-status-cron-job-29148654-pxk5m          0/1    Completed
0          3m41s
zen-route-custom-cert-refresh-cronjob-29147520-v5t57          0/1    Completed
0          18h
zen-route-custom-cert-refresh-cronjob-29148000-bkq5s          0/1    Completed
0          10h
zen-route-custom-cert-refresh-cronjob-29148480-zjpm5          0/1    Completed
0          177m
zen-svc-inst-status-cron-job-29148648-p5lwx          0/1    Completed
0          9m41s
zen-svc-inst-status-cron-job-29148652-mn7k6          0/1    Completed
0          5m41s
zen-svc-inst-status-cron-job-29148656-45ps5          0/1    Completed
0          101s
zen-watcher-b8ddddd56d-wm7tf          2/2    Running
0          9d

```

## Deploying Workflow Process Service Runtime

After configuring IBM Cloud Pak for Business Automation components, you can deploy Workflow Process Service Runtime.

1. If you want to use an EDB Postgres server, you can proceed directly to step 2. If you have an external PostgreSQL server, complete the following steps:
  - a. Prepare your PostgreSQL database. To make sure the PostgreSQL database is configured correctly for your workload, update the following parameters in the `postgresql.conf` file of the database server:

Parameter	Setting	Description
<code>shared_buffers</code>	Minimum 1024 MB	PostgreSQL performance tuning recommends using 25% of the memory for the shared buffer. Updates to the Linux kernel configuration might also be required. For more information, see the PostgreSQL tuning guides.
<code>work_mem</code>	Minimum 20 MB	This parameter applies to each session and many user sessions can cause large memory usage. This memory is critical because it is used for sort operations. The running time can increase significantly if the value is set too low. For example, the running time might be over an hour for toolkit deployments.
<code>max_prepared_transactions</code>	For example, 200	This value should be at least as large as the <code>max_connections</code> setting.

Parameter	Setting	Description
max_wal_size	For example, 6 GB	For larger workloads, the default value must be increased. To check whether an increase is required, see the PostgreSQL server log files.
log_min_duration_statement	For example, 5000	This optional parameter is for logging statements that exceed the specified running time to identify bottlenecks and potential tuning areas. This value is measured in milliseconds. For example, a value of 5000 corresponds to 5 seconds.

- b. Update the values for `database.external.databaseName`, `database.external.dbCredentialSecret`, and `database.external.dbServerCertSecret`. For more information about other database parameters, see [Workflow Process Service parameters](#).
- c. Create a database in your external PostgreSQL server and create a user secret, where username corresponds to the database username, and password corresponds to the database password. If you want to enable certificate-based authentication, you do not need a password for `wfps-db-secret`. For example, your file might look similar to:

```
apiVersion: v1
kind: Secret
metadata:
  name: wfps-db-secret
type: Opaque
stringData:
  username: "wfpsadmin"
  password: "password"
```

- d. By default, SSL communication is enabled. If you want to disable SSL, change the value of `database.external.enableSSL` to `false`.

If you want to enable SSL, create a CA certificate secret with the `ca.crt` key, by using the `ca.crt` file that is exported from your PostgreSQL server. For the secret name, enter the value of `database.external.dbServerCertSecret`. For example, if you are enabling SSL by itself, the command might look similar to:

```
kubectl create secret generic wfps-db-cacert-secret --from-file=ca.crt=./ca.crt.pem
```

Your database configuration might look similar to:

```
spec:
  database:
    external:
      type: postgresql
      enableSSL: true
      dbServerCertSecret: wfps-db-cacert-secret
```

Optionally, if you want to enable both SSL and database certificate-based authentication, create the secret with `client.crt` and `client.key`. Set the value of `spec.database.external.sslMode` to `verify-ca` or `verify-full`. To create your secret, run a command similar to:

```
kubectl create secret generic wfps-db-cacert-secret --from-file=ca.crt=./ca.crt.pem --from-file=tls.crt=./client.crt --from-file=tls.key=./client.key
```

- e. Optional: If you want to use custom Java™ database connectivity (JDBC) files inside the Workflow Process Service Runtime server, set the `database.customJDBCPVC` parameter. The persistent volume claim (PVC) needs to be in ROX (ReadOnlyMany) or RWX (ReadWriteMany) access mode, otherwise high availability disaster recovery (HADR) will be affected, since all pods must be allocated to the same node and the PVC is mounted at the `/shared/resources/jdbc/postgresql` directory inside the container. The `jdbc/postgresql` directory must be created inside `customJDBCPVC`. For example, the structure of the remote file system might look like:

```
jdbc
├── postgresql
│   └── postgresql-42.2.15.jar
```

2. Prepare storage for the Workflow Process Service Runtime server. The Workflow Process Service Runtime server supports persistence on three types of files: data files, log files and dump files. By default, data file persistence is enabled, while log and dump file persistence are disabled.
- Option 1: If your environment supports dynamic provisioning, you can enable or disable each type of file persistence and configure each storage class. For example:

```
spec:
  persistent:
    data:
      enable: true
      size: 1Gi
      storageClassName: data-storage-class
    dump:
      enable: false
      size: 1Gi
      storageClassName: dump-storage-class
    logs:
      enable: false
      size: 1Gi
      storageClassName: logs-storage-class
```


- Option 2: If your environment does not support dynamic provisioning and you do not have available storage classes on your cluster, choose this option. You can manually create your PV with `spec.storageClass` configured so that it can be used as `storageClassName` in the Workflow Process Service Runtime custom resource. If you want to scale up to 2 instances of the Workflow Process Service Runtime server, you need to create another set of PVs and associate them with corresponding storage class.
3. Use the `shared_configuration.sc_generate_sample_network_policies` parameter in the Cloud Pak for Business Automation custom resource to control internet access. The default value is false. You must set it to true to create sample network policies, which can be installed after the CP4BA deployment is ready by running a script. For more information, see .
4. Create a custom resource YAML file for your Workflow Process Service Runtime configuration. For more information about parameters, see [Workflow Process Service parameters](#) . After you complete the following steps, your custom resource might look similar to the following:

```
apiVersion: icp4a.ibm.com/v1
kind: WfPSRuntime
metadata:
  name: wfps-instance1
spec:
  appVersion: "25.0.0"
  deploymentLicense: production
  license:
    accept: true
```

- a. Optional: If you are using LDAP, it is recommended to update the value of `spec.admin.username` with an LDAP user.

By default, the operator sets `spec.admin.username` to be the Common Services admin user from the `platform-auth-idp-credential` secret in the Common Services namespace.

- If you are using the shared Common Services, the namespace is `ibm-common-services`.

- If you are using a dedicated Common Services, you can find the namespace from the common-service-maps ConfigMap from the kube-public namespace. For more information about the common-service-maps ConfigMap, see step 2 in [Setting up the cluster in the OpenShift console](#) .

You can configure LDAP in Identity Access Management and then set the LDAP user to be `spec.admin.username`. The Workflow Process Service Runtime operator automatically configures the LDAP user as a Zen user. To configure LDAP, see step 1 of [“Completing post-deployment tasks for Workflow Process Service Runtime”](#) on page 43.

- If you want to let the Workflow Process Service Runtime operator provision an EDB Postgres instance, you must make sure that your OpenShift Container Platform cluster already has a default storage class defined. If there is no default storage class that is defined, set the storage class name by using the `spec.persistent.storageClassName` parameter. For example:

```
spec:
  persistent:
    storageClassName: <storage_class_name>
```

- Optional: If you want to add custom files inside the Workflow Process Service Runtime server, you can update `node.customFilePVC`. The persistent volume claim (PVC) must be in ROX (ReadOnlyMany) or RWX (ReadWriteMany) access mode, otherwise HADR is affected, since all pods must be allocated to the same node and the PVC is mounted at the `/opt/ibm/bawfile` directory inside the container. For example, the `customFilePVC` might look similar to:

```
spec:
  node:
    customFilePVC: my-custom-wfps-pvc
```

- Optional: If you want to enable the full text search feature, include the following lines:

```
spec:
  capabilities:
    fullTextSearch:
      enable: true
      adminGroups:
        - example_group
    esStorage:
      storageClassName: BlockStorageClassName
      size: 50Gi
    esSnapshotStorage:
      storageClassName: BlockStorageClassName
      size: 10Gi
```

If you installed Cloud Pak foundational services Elasticsearch, you don't need to add `capabilities.fullTextSearch.esStorage` and `capabilities.fullTextSearch.esSnapshotStorage`. If you didn't install Cloud Pak foundational services Elasticsearch and `capabilities.fullTextSearch.enable` is set to true, you must add `capabilities.fullTextSearch.esStorage` and `capabilities.fullTextSearch.esSnapshotStorage` in the custom resource yaml file. The StorageClass for Elasticsearch and Elasticsearch snapshot should create the storage type of the PVs in block mode rather than file system mode.

- Optional: If you want to start external services in your workflows, you need to retrieve the certificate of the external service and add it into the server trust list as a secret. For the `serverTrustCertificateList` parameter, enter a list of secrets, where every secret stores a trusted certificate. To create a secret, run the following command:

```
kubectl create secret generic <example_ocp_external_default_certificate> --from-file=tls.crt=./cert.crt
```

Your `serverTrustCertificateList` configuration might look similar to:

```
spec:
  tls:
    serverTrustCertificateList:
      - <example_ocp_external_default_certificate>
```

f. Apply the custom resource by running the following command:

```
oc apply -f <custom_resource_name>.yaml
```

g. After a few minutes, verify that you see your pods, services, and route. If you chose EDB Postgres, the server pod and services are also listed. For example:

```
[root@xxxxxxx]# oc get pod
wfps-instance1-postgre-0 1/1 Running 0 21h
wfps-instance1-wfps-runtime-server-0 1/1 Running 0 21h
Running 0 21h 1/1
```

```
[root@xxxxxxx]# oc get service
NAME                                     TYPE          CLUSTER-IP
EXTERNAL-IP  PORT(S)      AGE
wfps-instance1-postgre-any             ClusterIP      172.30.60.216
<none>          5432/TCP      6d
wfps-instance1-postgre-r               ClusterIP      172.30.43.94
<none>          5432/TCP      6d
wfps-instance1-postgre-ro              ClusterIP      172.30.234.237
<none>          5432/TCP      6d
wfps-instance1-postgre-rw              ClusterIP      172.30.105.46
<none>          5432/TCP      6d
wfps-instance1-wfps-headless-service   ClusterIP      None
<none>          9443/TCP      6d
wfps-instance1-wfps-service
```

```
[root@xxxxxxx]# oc get route
NAME      HOST/PORT          WILDCARD   PATH   SERVICES
PORT      TERMINATION
cpd       cpd-cp4a-project.apps.xxxxxx.cp.fyre.ibm.com  ibm-nginx-svc  ibm-nginx-
https-port passthrough/Redirect  None
```

## Completing post-deployment tasks for Workflow Process Service Runtime

You can choose to configure LDAP or a third-party identity access and management provider.

1. Go to the cluster in your Common UI console. To access the Common UI console, see [Accessing your cluster by using the console](#). To configure an LDAP connection, see [Configuring LDAP connection](#).
2. Add LDAP users in Cloud Pak Platform UI.

a. Connect to the URL: `https://cluster_address`, where `cluster_address` is the IBM Cloud Pak console route. You can get the IBM Cloud Pak console route by running the command:

```
oc get route cpd -o jsonpath='{.spec.host}' && echo
```

The output might look similar to:

```
cpd-namespace_name.apps.mycluster.mydomain
```

Using the example output, the console URL would look similar to:

```
https://cpd-namespace_name.apps.mycluster.mydomain/zen
```

- b. Log in to the IBM Cloud Pak dashboard and select OpenShift Container Platform authentication for kubeadmin, or log in with the IBM provided credentials from step 1a, if you are an administrator.
- c. Go to **Manage users > Add users**.
- d. Type the names of users that you want to add, and click **Next**.
- e. Assign the users to roles, or add them to a group. You can add your LDAP user under **Users** or you can add your LDAP user group under **User groups**. For both users and user groups, make sure that at least one role is selected. For example, roles include administrator, automation administrator, automation analyst, automation developer, automation operator, and user.
- f. Click **Add** to register the users.

## Verifying your Workflow Process Service Runtime deployment

To access services provided by Workflow Process Service Runtime, you might need to log in with your LDAP user and password.

1. Make sure your Workflow Process Service Runtime deployment is ready by running the command:

```
oc get wfps <CR_name> -o=jsonpath='{.status.components.wfps.configurations[*].value}'
```

The output might look similar to:

```
<CR_name>-admin-client-secret Ready Ready Ready Ready
```

2. To access the Workplace console, you have two options. You can run the command:

```
oc get wfps <CR_name> -o=jsonpath='{.status.endpoints[2].uri}'
```

Alternatively, you can manually splice the Workplace console URL:

```
https://(oc get route cpd -o jsonpath='{.spec.host}')
```

```
<CR_name>-wfps/Workplace
```

For example, the resulting Workplace console URL might look like:

```
https://cpd-cp4a-project.apps.xxxxxx.cp.fyre.ibm.com/<CR_name>-wfps/Workplace
```

3. To access the Operations REST APIs Swagger UI, you have two options. You can run the command:

```
oc get wfps <CR_name> -o=jsonpath='{.status.endpoints[3].uri}'
```

Alternatively, you can manually splice the Operations REST APIs Swagger UI URL:

```
https://(oc get route cpd -o jsonpath='{.spec.host}')
```

```
<CR_name>-wfps/ops/explorer
```

For example, the resulting Operations REST APIs Swagger UI URL might look like:

```
https://cpd-cp4a-project.apps.xxxxxx.cp.fyre.ibm.com/<CR_name>-wfps/ops/explorer
```

4. To construct the URLs of exposed REST services and exposed web services, you must locate the endpoint of Workflow Process Service Runtime in the custom resource file's status field. To determine the URL of your REST services and web services, complete the following steps:

- a. Run the command:

```
oc get wfps wfps-instance1 -o yaml
```

- b. In the endpoints section, locate the URI of the external Workflow Process Service Runtime instance. For example:

```
- name: External Base URL
  scope: External
  type: https
  uri: https://cpd-wfps3.apps.fjk-ocp474.cp.example.com/wfps-instance1-wfps
```

- c. The URLs of your REST services have the following structure:

```
http://host_name:port/[<custom_prefix>]/automation-services/rest/<process_app_name>/[<snapshot_name>]/<rest_service_name>/docs{}
```

Where:

- `https://host_name:port/[<custom_prefix>]/` is your URI value from the previous step.
- `<process_app_name>` is the name of the process application.
- `<snapshot_name>` is the optional name of the snapshot.
- `<rest_service_name>` is the name of the REST service.

- d. The URL of your web services have the following structure:


```
https://host_name:port/[<custom_prefix>/]teamworks/webservices/<process_app_name>/  
[<snapshot_name>/]<web_service_name>.tw
```

Where:

- `https://host_name:port/[<custom_prefix>/]` is your URI value from step 4b.
- `<process_app_name>` is the name of the process application.
- `<snapshot_name>` is the optional name of the snapshot.
- `<web_service_name>` is the name of the web service.

## Managing your EDB Postgres server

### 1. To access data in your EDB Postgres server:

- a. Run the command `oc get cluster` to get the EDB Postgres cluster name. For example, the cluster name might be similar to `wfps-instance1-postgre`.
- b. Run the command `kubect1 port-forward --address 0.0.0.0 wfps-instance1-postgre-rw 5432:5432` on the OpenShift Container Platform infrastructure node. The infrastructure node IP and port (5432) are the database server and database port that are externally accessible.
- c. Get the username and password from the `wfps-instance1-postgre-app` secret to access the default database `wfpsdb`. To expose more EDB Postgres services, see [Exposing Postgres Services](#) .




### 2. Check your license.

- a. Run the command `oc get cluster` to get the EDB Postgres cluster name. For example, the cluster name might be similar to `wfps-instance1-postgre`.
- b. Run the command `oc get cluster <cluster_name> -o yaml` to check the license status. The output might look like:

```
licenseStatus:  
  isTrial: true  
  licenseExpiration: "2024-10-01T00:00:00Z"  
  licenseStatus: Valid license (IBM - Data & Analytics (Cloud))  
  repositoryAccess: false  
  valid: true
```

### 3. To configure backup and recovery for EDB Postgres, see [Backup and Recovery](#) .

### 4. You can configure the operator's management of the EDB Postgres cluster.

- a. If you want to manage the EDB Postgres cluster, you need to update the value of `spec.database.managed.managementState` to `Unmanaged` in the Workflow Process Service Runtime custom resource yaml file. After you update the value of `spec.database.managed.managementState`, the Workflow Process Service Runtime operator will not manage the EDB Postgres cluster. To change the parameters and resources of the EDB Postgres cluster, see [PostgreSQL Configuration](#)  and [Resource management](#) . To add `nodeSelector` and select the nodes that a pod can run on, see [Node selection through nodeSelector](#) .

When you are in the `Unmanaged` state, you need to manually delete the EDB Postgres cluster after you delete the Workflow Process Service Runtime instance. For example, to delete your cluster, your command might look similar to:

```
oc delete cluster wfps-instance1-postgre
```

where `wfps-instance1-postgre` is the name of your EDB Postgres cluster.

- b. If you want the Workflow Process Service Runtime operator to manage the EDB Postgres cluster, set `spec.database.managed.managementState` to `Managed`. The EDB Postgres cluster will have the default configuration and the EDB Postgres cluster will be deleted automatically after the Workflow Process Service Runtime instance is deleted.

## Optional: Configuring Workflow Process Service Runtime to work with Workflow Process Service Authoring

If you are using Workflow Process Service Authoring, you can configure Workflow Process Service Runtime to debug and test instances from Workflow Process Service Authoring on a production or test environment.

Depending on the capabilities that you deploy with Workflow Process Service Authoring, you can see your tasks in different places.

If Workflow Process Service Runtime and Workflow Process Service Authoring are deployed in the same namespace:

- For Workflow Process Service Runtime, Workplace in IBM Business Automation Navigator connects to an external Process Federation Server. As a result, you see runtime tasks through Workplace.
- For Workflow Process Service Authoring, the authoring server uses the embedded Process Federation Server, so you see authoring tasks through the embedded Workplace.

### About this task

To configure Workflow Process Service Runtime to work with Workflow Process Service Authoring:

### Procedure

1. Exchange the Cloud Pak Platform UI Certificate Authority (CA) of Workflow Process Service Runtime and Workflow Process Service Authoring. If you installed them under the same namespace, you can go directly to step 2.

For detailed instructions about extracting a CA, see [Exporting the Zen CA and common services CA](#).

- a. Extract the Cloud Pak Platform UI (Zen) CA from your authoring installation, and copy it to your runtime container. Create a secret with the CA.

For example, if the location of the CA in your runtime environment is `/root/zenRootCAPC.cert`, you can run the following command to create a secret:

```
kubectl create secret generic wfps-tls-zen-secret --from-file=tls.crt=/root/zenRootCAPC.cert
```

- b. Extract the Cloud Pak Platform UI (Zen) CA from your runtime installation, and copy it to your authoring container. Create a secret with the CA.

For example, if the location of the CA in your authoring environment is `/root/zenRootCAPS.cert` and `/root/csRootCAPS.cert`, you can run the following command to create secrets:

```
kubectl create secret generic wfpsaut-tls-zen-secret --from-file=tls.crt=/root/zenRootCAPS.cert
kubectl create secret generic wfpsaut-tls-cs-secret --from-file=tls.crt=/root/csRootCAPS.cert
```

2. Exchange the router-ca certificates between the runtime and authoring environment.

- a. In the runtime environment, run the following command:

```
oc get secret router-ca -n openshift-ingress-operator -o template --
template='{{ index .data "tls.crt" }}' | base64 --decode > routercaPS.cert
```

Copy the certificate to the authoring environment and create the secret.

```
oc create secret generic wfpsaut-routerca-secret --from-file=tls.crt=routercaPS.cert
```



Then add the certificate to the trust list in the authoring custom resource (CR).

- b. In the authoring environment, run the following command:

```
oc get secret router-ca -n openshift-ingress-operator -o template --
template='{{ index .data "tls.crt" }}' | base64 --decode > routercaPC.cert
```

Copy the certificate to the runtime environment and create the secret.

```
oc create secret generic wfps-routerca-secret --from-file=tls.crt=routercaPC.cert
```

Then add the certificate to the trust list in the runtime custom resource.

3. In the Workflow Process Service Runtime environment, create a secret that is named `ibm-wfps-wc-secret` that contains the username and password of the Workflow Process Service administrator. Add the secret to the `adminSecrets4operator-ctnrs.yaml` file.

For example, the section might look similar to:

```
apiVersion: v1
kind: Secret
metadata:
  name: ibm-wfps-wc-secret
type: Opaque
stringData:
  username: <wfps_authoring_admin_user>
  password: <wfps_authoring_admin_user_password>
```

Where `<workflow_authoring_admin_user>` is an admin user of Workflow Process Service Authoring.

If you did not configure Lightweight Directory Access Protocol (LDAP) or a third party provider, Identity Access Management is used for authentication. You can get the admin user information from the secret `ibm-iam-bindinfo-platform-auth-idp-credentials`.

If you configured LDAP, you can get the configuration from the `bastudio_configuration.admin_user` parameter in your Workflow Process Service Authoring custom resource (CR).

4. Apply the `adminSecrets4operator-ctnrs.yaml` file in the runtime environment by running the following command:

```
kubectl apply -f ./adminSecrets4operator-ctnrs.yaml
```

5. Add or update the Workflow Process Service Runtime CR YAML file.

- a. Add the Zen CA from Workflow Process Service Authoring that you copied in step 1 to the trust list section. If you installed the runtime and authoring environment in the same namespace, you can skip this step.
- b. Create the egress network policy for Workflow Process Service Runtime. If you need to install network policies for your deployment, you can create egress network policies with a single network policy or an individual network policy for the Workflow Process Service Runtime capability. For more information, see [Network policies to manage access to external services \(egress\)](#). If you are creating a custom "component-level" egress network policy, set `podSelector.matchLabels.com.ibm.cp4a.networking/egress-external-app-component` to `WfPS`. For example, your network policy might look similar to:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: "{{ meta.name }}"-wfps-runtime-egress-external-app"
  namespace: "{{ meta.namespace }}"
  labels:
    app.kubernetes.io/name: "{{ meta.name }}"-wfps-runtime-egress-external-app"
spec:
  podSelector:
    matchLabels:
      com.ibm.cp4a.networking/egress-external-app-component: "WfPS"
  policyTypes:
    - Egress
```

```

egress:
- to:
  - ipBlock:
      cidr: # IP address your external application . For example: 1.2.3.4/32
    ports:
      - protocol: #Protocol UDP or TCP
        port: #Port number

```

- c. Add or update the CR to include Workflow Process Service Authoring configuration. For example, the CR might have a new section that looks similar to:

```

apiVersion: icp4a.ibm.com/v1
kind: WfPSRuntime
metadata:
  name: wfps-instance1
spec:
  .....

  tls:
    serverTrustCertificateList: [wfps-tls-zen-secret, wfps-routerca-secret]

  authoringServer:
    url: "https://<bastudio_url>/bas/ProcessCenter"
    secretName: "ibm-wfps-wc-secret"
    heartbeatInterval: 30
    webPDUUrl: "https://<bastudio_url>/bas/WebPD"

```

To get bastudio\_url, you can run the following command:

```
kubectl describe configmaps icp4adeploy-cp4ba-access-info
```

6. Add or update the Workflow Process Service Authoring CR YAML file.
  - a. Add the Zen CA from Workflow Process Service Runtime that you copied in step 1 to the trust list section. If you installed the runtime and authoring environment in the same namespace, you can skip this step.
  - b. Create the egress network policy for Workflow Process Service Authoring. If you need to install network policies for your deployment, you can create egress network policies with a single network policy or an individual network policy for the Workflow Process Service Authoring capability. For more information, see [Network policies to manage access to external services \(egress\)](#). If you are creating a custom "component-level" egress network policy, set podSelector.matchLabels.com.ibm.cp4a.networking/egress-external-app-component to BAS.
  - c. Add or update the CR to include Workflow Process Service Runtime configuration. For example, the CR might have a new section that looks similar to:

```

bastudio_configuration:
  tls:
    tlsTrustList: [wfpsaut-tls-zen-secret, wfpsaut-tls-cs-secret, wfpsaut-routerca-secret]

  workflow_authoring_configuration:
    environment_config:
      # Content security policy additional directive for all folders , e.g. ["https://hostname1","https://hostname2"]
      content_security_policy_additional_all: [https://<wfps_runtime_url>/wfps-instance1-wfps/]

    csrf:
      ## Acceptable values in the Origin header field. The value of this property must be a comma-separated list of prefixes in the format protocol://host:port, e.g "https://example.com, http://example2.com:8080".
      origin_allowlist: "https://<cloudpak_url_for_business_automation_wfps_runtime>,https://<iam_url_for_business_automation_wfps_runtime>"
      ## Acceptable values in the Referer header field. The value of this property must be a comma-separated list of fully qualified host names, e.g "example1.com, example2.com".
      referer_allowlist: "<cloudpak_url_for_business_automation_wfps_runtime>,<iam_url_for_business_automation_wfps_runtime>"

```

Where:

- `<wfps_runtime_url>` can be found by running `kubectl get routes`. For example the URL might look similar to `cpd-test1.apps.mp88.cp.fyre.ibm.com`.
- `<cloudpak_url_for_business_automation_wfps_runtime>` can be found by running the command `oc get routes|grep cpd`
- `<iam_url_for_business_automation_wfps_runtime>` can be found by running the command `oc get routes|grep cp-console`

7. Apply the CR changes by running the following command:

```
kubectl apply -f <customResourceFileName>
```

8. Add users to the debug user group. For information about adding users, see [Managing access to the repository for workflow business automations](#).

