Installing and configuring OpenShift Pipelines
Abstract

This document provides information about installing and configuring OpenShift Pipelines.
# Table of Contents

## CHAPTER 1. INSTALLING OPENSOURCE PIPELINES

1. Prerequisites ........................................................................................................... 3
2. INSTALLING THE RED HAT OPENSOURCE PIPELINES OPERATOR IN WEB CONSOLE ........................................... 3
3. INSTALLING THE OPENSOURCE PIPELINES OPERATOR USING THE CLI .............................................................. 5
4. OPENSOURCE PIPELINES OPERATOR IN A RESTRICTED ENVIRONMENT ......................................................... 6
5. PERFORMANCE TUNING USING TEKTONCONFIG CR ..................................................................................... 6
6. ADDITIONAL RESOURCES ........................................................................................................... 8

## CHAPTER 2. UNINSTALLING OPENSOURCE PIPELINES

1. DELETING THE RED HAT OPENSOURCE PIPELINES COMPONENTS AND CUSTOM RESOURCES ..... 9
2. UNINSTALLING THE RED HAT OPENSOURCE PIPELINES OPERATOR ............................................................ 9

## CHAPTER 3. CUSTOMIZING CONFIGURATIONS IN THE TEKTONCONFIG CUSTOM RESOURCE

1. PREREQUISITES ........................................................................................................... 11
2. CONFIGURING THE RED HAT OPENSOURCE PIPELINES CONTROL PLANE ......................................................... 11
3. Modifiable fields with default values .................................................................................. 12
4. Optional configuration fields ............................................................................................. 13
5. CHANGING THE DEFAULT SERVICE ACCOUNT FOR OPENSOURCE PIPELINES .................................................. 14
6. DISABLING THE SERVICE MONITOR .................................................................................. 14
7. CONFIGURING PIPELINE RESOLVERS ............................................................................... 14
8. DISABLING CLUSTER TASKS AND PIPELINE TEMPLATES .................................................................................. 15
9. DISABLING THE INTEGRATION OF TEKTON HUB .................................................................................. 16
10. AUTOMATIC CREATION OF RBAC RESOURCES .................................................................................. 16
11. AUTOMATIC PRUNING OF TASK RUNS AND PIPELINE RUNS ........................................................................ 17
   1. Configuring the pruner .................................................................................................. 17
   2. Annotations for automatically pruning task runs and pipeline runs ................................. 18
12. ADDITIONAL RESOURCES ................................................................................................. 19
CHAPTER 1. INSTALLING OPENSHIFT PIPELINES

This guide walks cluster administrators through the process of installing the Red Hat OpenShift Pipelines Operator to an OpenShift Container Platform cluster.

Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with `cluster-admin` permissions.
- You have installed `oc` CLI.
- You have installed OpenShift Pipelines (`tkn`) CLI on your local system.
- Your cluster has the Marketplace capability enabled or the Red Hat Operator catalog source configured manually.

1.1. INSTALLING THE RED HAT OPENSHIFT PIPELINES OPERATOR IN WEB CONSOLE

You can install Red Hat OpenShift Pipelines using the Operator listed in the OpenShift Container Platform OperatorHub. When you install the Red Hat OpenShift Pipelines Operator, the custom resources (CRs) required for the pipelines configuration are automatically installed along with the Operator.

The default Operator custom resource definition (CRD) `config.operator.tekton.dev` is now replaced by `tektonconfigs.operator.tekton.dev`. In addition, the Operator provides the following additional CRDs to individually manage OpenShift Pipelines components: `tektonpipelines.operator.tekton.dev`, `tektontriggers.operator.tekton.dev` and `tektonaddons.operator.tekton.dev`.

If you have OpenShift Pipelines already installed on your cluster, the existing installation is seamlessly upgraded. The Operator will replace the instance of `config.operator.tekton.dev` on your cluster with an instance of `tektonconfigs.operator.tekton.dev` and additional objects of the other CRDs as necessary.

WARNING

If you manually changed your existing installation, such as, changing the target namespace in the `config.operator.tekton.dev` CRD instance by making changes to the `resource name - cluster` field, then the upgrade path is not smooth. In such cases, the recommended workflow is to uninstall your installation and reinstall the Red Hat OpenShift Pipelines Operator.

The Red Hat OpenShift Pipelines Operator now provides the option to choose the components that you want to install by specifying profiles as part of the `TektonConfig` custom resource (CR). The `TektonConfig` CR is automatically installed when the Operator is installed. The supported profiles are:

- Lite: This installs only Tekton Pipelines.
- Basic: This installs Tekton Pipelines, Tekton Triggers, and Tekton Chains.
All: This is the default profile used when the TektonConfig CR is installed. This profile installs all of the Tekton components, including Tekton Pipelines, Tekton Triggers, Tekton Chains, Pipelines as Code, and Tekton Addons. Tekton Addons includes the ClusterTasks, ClusterTriggerBindings, ConsoleCLIDownload, ConsoleQuickStart, and ConsoleYAMLSample resources.

Procedure

1. In the Administrator perspective of the web console, navigate to Operators → OperatorHub.

2. Use the Filter by keyword box to search for Red Hat OpenShift Pipelines Operator in the catalog. Click the Red Hat OpenShift Pipelines Operator tile.

3. Read the brief description about the Operator on the Red Hat OpenShift Pipelines Operator page. Click Install.

4. On the Install Operator page:

   a. Select All namespaces on the cluster (default) for the Installation Mode. This mode installs the Operator in the default openshift-operators namespace, which enables the Operator to watch and be made available to all namespaces in the cluster.

   b. Select Automatic for the Approval Strategy. This ensures that the future upgrades to the Operator are handled automatically by the Operator Lifecycle Manager (OLM). If you select the Manual approval strategy, OLM creates an update request. As a cluster administrator, you must then manually approve the OLM update request to update the Operator to the new version.

   c. Select an Update Channel

      • The latest channel enables installation of the most recent stable version of the Red Hat OpenShift Pipelines Operator. Currently, it is the default channel for installing the Red Hat OpenShift Pipelines Operator.

      • To install a specific version of the Red Hat OpenShift Pipelines Operator, cluster administrators can use the corresponding pipelines-<version> channel. For example, to install the Red Hat OpenShift Pipelines Operator version 1.8.x, you can use the pipelines-1.8 channel.

      NOTE

      Starting with OpenShift Container Platform 4.11, the preview and stable channels for installing and upgrading the Red Hat OpenShift Pipelines Operator are not available. However, in OpenShift Container Platform 4.10 and earlier versions, you can use the preview and stable channels for installing and upgrading the Operator.

5. Click Install. You will see the Operator listed on the Installed Operators page.

   NOTE

   The Operator is installed automatically into the openshift-operators namespace.

6. Verify that the Status is set to Succeeded Up to date to confirm successful installation of Red Hat OpenShift Pipelines Operator.
7. Verify that all components of the Red Hat OpenShift Pipelines Operator were installed successfully. Login to the cluster on the terminal, and run the following command:

$ oc get tektonconfig config

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>READY</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>tektonpipeline.operator.tekton.dev/pipeline</td>
<td>v0.47.0</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>

If the READY condition is **True**, the Operator and its components have been installed successfully.

Additionally, check the components’ versions by running the following command:

$ oc get tektonpipeline,tektontrigger,tektonchain,tektonaddon,pac

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>READY</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>tektonpipeline.operator.tekton.dev/pipeline</td>
<td>v0.47.0</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>tektontrigger.operator.tekton.dev/trigger</td>
<td>v0.23.1</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>tektonchain.operator.tekton.dev/chain</td>
<td>v0.16.0</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>tektonaddon.operator.tekton.dev/addon</td>
<td>1.11.0</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>openshiftpipelinesascode.operator.tekton.dev/pipelines-as-code</td>
<td>v0.19.0</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>

### 1.2. INSTALLING THE OPENSHIFT PIPELINES OPERATOR USING THE CLI

You can install Red Hat OpenShift Pipelines Operator from the OperatorHub using the CLI.

**Procedure**
1. Create a Subscription object YAML file to subscribe a namespace to the Red Hat OpenShift Pipelines Operator, for example, `sub.yaml`:

**Example Subscription**

```yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-pipelines-operator
  namespace: openshift-operators
spec:
  channel: <channel name>  
  name: openshift-pipelines-operator-rh
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

1. The channel name of the Operator. The `pipelines-<version>` channel is the default channel. For example, the default channel for Red Hat OpenShift Pipelines Operator version 1.7 is `pipelines-1.7`. The `latest` channel enables installation of the most recent stable version of the Red Hat OpenShift Pipelines Operator.

2. Name of the Operator to subscribe to.

3. Name of the CatalogSource that provides the Operator.

4. Namespace of the CatalogSource. Use `openshift-marketplace` for the default OperatorHub CatalogSources.

2. Create the Subscription object:

```
$ oc apply -f sub.yaml
```

The subscription installs the Red Hat OpenShift Pipelines Operator into the `openshift-operators` namespace. The Operator automatically installs OpenShift Pipelines into the default `openshift-pipelines` target namespace.

### 1.3. RED HAT OPENShift PIPELINES OPERATOR IN A RESTRICTED ENVIRONMENT

The Red Hat OpenShift Pipelines Operator enables support for installation of pipelines in a restricted network environment.

The Operator installs a proxy webhook that sets the proxy environment variables in the containers of the pod created by tekton-controllers based on the `cluster` proxy object. It also sets the proxy environment variables in the TektonPipelines, TektonTriggers, Controllers, Webhooks, and Operator Proxy Webhook resources.

By default, the proxy webhook is disabled for the `openshift-pipelines` namespace. To disable it for any other namespace, you can add the `operator.tekton.dev/disable-proxy: true` label to the namespace object.

### 1.4. PERFORMANCE TUNING USING TEKTONCONFIG CR
You can modify the fields under the `.spec.pipeline.performance` parameter in the TektonConfig custom resource (CR) to change high availability (HA) support and performance configuration for the OpenShift Pipelines controller.

**Example TektonConfig performance fields**

```yaml
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  pipeline:
    performance:
      disable-ha: false
      buckets: 1
      threads-per-controller: 2
      kube-api-qps: 5.0
      kube-api-burst: 10
```

The fields are optional. If you set them, the Red Hat OpenShift Pipelines Operator includes most of the fields as arguments in the `openshift-pipelines-controller` deployment under the `openshift-pipelines-controller` container. The OpenShift Pipelines Operator also updates the `buckets` field in the `config-leader-election` configuration map under the `openshift-pipelines` namespace.

If you do not specify the values, the OpenShift Pipelines Operator does not update those fields and applies the default values for the OpenShift Pipelines controller.

**NOTE**

If you modify or remove any of the performance fields, the OpenShift Pipelines Operator updates the `openshift-pipelines-controller` deployment and the `config-leader-election` configuration map (if the `buckets` field changed) and re-creates `openshift-pipelines-controller` pods.

### Table 1.1. Modifiable fields for tuning OpenShift Pipelines performance

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default value for the OpenShift Pipelines controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>disable-ha</td>
<td>Enable or disable the high availability (HA) support. By default, the HA support is enabled.</td>
<td>false</td>
</tr>
<tr>
<td>buckets</td>
<td>The number of buckets used to partition the key space for each reconciler.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Each of the replicas uses these buckets. The instance that owns a bucket reconciles the keys partitioned into that bucket. The maximum value is 10</td>
<td></td>
</tr>
</tbody>
</table>
The number of threads (workers) to use when the work queue of the OpenShift Pipelines controller is processed.

**kube-api-qps**
The maximum queries per second (QPS) to the cluster master from the REST client.

**kube-api-burst**
The maximum burst for a throttle.

---

**NOTE**
The OpenShift Pipelines Operator does not control the number of replicas of the OpenShift Pipelines controller. The `replicas` setting of the deployment determines the number of replicas. For example, to change the number of replicas to 3, enter the following command:

```
$ oc --namespace openshift-pipelines scale deployment openshift-pipelines-controller --replicas=3
```

**IMPORTANT**
The `kube-api-qps` and `kube-api-burst` fields are multiplied by 2 in the OpenShift Pipelines controller. For example, if the `kube-api-qps` and `kube-api-burst` values are 10, the actual QPS and burst values become 20.

---

### 1.5. ADDITIONAL RESOURCES

- You can learn more about installing Operators on OpenShift Container Platform in the adding Operators to a cluster section.
- To install Tekton Chains using the Red Hat OpenShift Pipelines Operator, see Using Tekton Chains for Red Hat OpenShift Pipelines supply chain security.
- To install and deploy in-cluster Tekton Hub, see Using Tekton Hub with Red Hat OpenShift Pipelines.
- For more information on using pipelines in a restricted environment, see:
  - Mirroring images to run pipelines in a restricted environment
  - Configuring Samples Operator for a restricted cluster
  - Creating a cluster with a mirrored registry
Cluster administrators can uninstall the Red Hat OpenShift Pipelines Operator by performing the following steps:

1. Delete the Custom Resources (CRs) that were added by default when you installed the Red Hat OpenShift Pipelines Operator.

2. Delete the CRs of the optional components such as Tekton Hub that depend on the Operator.

   CAUTION

   If you uninstall the Operator without removing the CRs of optional components, you cannot remove them later.


Uninstalling only the Operator will not remove the Red Hat OpenShift Pipelines components created by default when the Operator is installed.

2.1. DELETING THE RED HAT OPENSHIFT PIPELINES COMPONENTS AND CUSTOM RESOURCES

Delete the Custom Resources (CRs) created by default during installation of the Red Hat OpenShift Pipelines Operator.

Procedure

1. In the Administrator perspective of the web console, navigate to Administration → Custom Resource Definition.

2. Type config.operator.tekton.dev in the Filter by name box to search for the Red Hat OpenShift Pipelines Operator CRs.

3. Click CRD Config to see the Custom Resource Definition Details page.

4. Click the Actions drop-down menu and select Delete Custom Resource Definition

   NOTE

   Deleting the CRs will delete the Red Hat OpenShift Pipelines components, and all the tasks and pipelines on the cluster will be lost.

5. Click Delete to confirm the deletion of the CRs.

   IMPORTANT

   Repeat the procedure to find and remove CRs of optional components such as Tekton Hub before uninstalling the Operator. If you uninstall the Operator without removing the CRs of optional components, you cannot remove them later.

2.2. UNINSTALLING THE RED HAT OPENSHIFT PIPELINES OPERATOR
You can uninstall the Red Hat OpenShift Pipelines Operator by using the Administrator perspective in the web console.

Procedure

1. From the Operators → OperatorHub page, use the Filter by keyword box to search for the Red Hat OpenShift Pipelines Operator.

2. Click the Red Hat OpenShift Pipelines Operator tile. The Operator tile indicates that the Operator is installed.


**WARNING**

When you uninstall the {pipeline-shortname} Operator, all resources within the target namespace where the Operator is installed, the openshift-pipelines namespace by default, are lost including the secrets you configured with it.

Additional resources

- You can learn more about uninstalling Operators on OpenShift Container Platform in the deleting Operators from a cluster section.
CHAPTER 3. CUSTOMIZING CONFIGURATIONS IN THE TEKTONCONFIG CUSTOM RESOURCE

In Red Hat OpenShift Pipelines, you can customize the following configurations by using the TektonConfig custom resource (CR):

- Configuring the Red Hat OpenShift Pipelines control plane
- Changing the default service account
- Disabling the service monitor
- Configuring pipeline resolvers
- Disabling cluster tasks and pipeline templates
- Disabling the integration of Tekton Hub
- Disabling the automatic creation of RBAC resources
- Pruning of task runs and pipeline runs

3.1. PREREQUISITES

- You have installed the Red Hat OpenShift Pipelines Operator.

3.2. CONFIGURING THE RED HAT OPENSSHIFT PIPELINES CONTROL PLANE

You can customize the OpenShift Pipelines control plane by editing the configuration fields in the TektonConfig custom resource (CR). The Red Hat OpenShift Pipelines Operator automatically adds the configuration fields with their default values so that you can use the OpenShift Pipelines control plane.

Procedure

1. In the Administrator perspective of the web console, navigate to Administration → CustomResourceDefinitions.

2. Use the Search by name box to search for the tektonconfigs.operator.tekton.dev custom resource definition (CRD). Click TektonConfig to see the CRD details page.

3. Click the Instances tab.

4. Click the config instance to see the TektonConfig CR details.

5. Click the YAML tab.

6. Edit the TektonConfig YAML file based on your requirements.

Example of TektonConfig CR with default values

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
```
3.2.1. Modifiable fields with default values

The following list includes all modifiable fields with their default values in the TektonConfig CR:

- **running-in-environment-with-injected-sidecars** (default: true): Set this field to false if pipelines run in a cluster that does not use injected sidecars, such as Istio. Setting it to false decreases the time a pipeline takes for a task run to start.

  **NOTE**
  
  For clusters that use injected sidecars, setting this field to false can lead to an unexpected behavior.

- **await-sidecar-readiness** (default: true): Set this field to false to stop OpenShift Pipelines from waiting for TaskRun sidecar containers to run before it begins to operate. This allows tasks to be run in environments that do not support the downwardAPI volume type.

- **default-service-account** (default: pipeline): This field contains the default service account name to use for the TaskRun and PipelineRun resources, if none is specified.

- **require-git-ssh-secret-known-hosts** (default: false): Setting this field to true requires that any Git SSH secret must include the known_hosts field.

  - For more information about configuring Git SSH secrets, see Configuring SSH authentication for Git in the Additional resources section.

- **enable-tekton-oci-bundles** (default: false): Set this field to true to enable the use of an experimental alpha feature named Tekton OCI bundle.

- **enable-api-fields** (default: stable): Setting this field determines which features are enabled. Acceptable value is stable, beta, or alpha.
Red Hat OpenShift Pipelines does not support the alpha value.

- **enable-provenance-in-status** (default: false): Set this field to true to enable populating the provenance field in TaskRun and PipelineRun statuses. The provenance field contains metadata about resources used in the task run and pipeline run, such as the source from where a remote task or pipeline definition was fetched.

- **enable-custom-tasks** (default: true): Set this field to false to disable the use of custom tasks in pipelines.

- **disable-creds-init** (default: false): Set this field to true to prevent OpenShift Pipelines from scanning attached service accounts and injecting any credentials into your steps.

- **disable-affinity-assistant** (default: true): Set this field to false to enable affinity assistant for each TaskRun resource sharing a persistent volume claim workspace.

**Metrics options**

You can modify the default values of the following metrics fields in the TektonConfig CR:

- **metrics.taskrun.duration-type** and **metrics.pipelinerun.duration-type** (default: histogram): Setting these fields determines the duration type for a task or pipeline run. Acceptable value is gauge or histogram.

- **metrics.taskrun.level** (default: task): This field determines the level of the task run metrics. Acceptable value is taskrun, task, or namespace.

- **metrics.pipelinerun.level** (default: pipeline): This field determines the level of the pipeline run metrics. Acceptable value is pipelinerun, pipeline, or namespace.

### 3.2.2. Optional configuration fields

The following fields do not have a default value, and are considered only if you configure them. By default, the Operator does not add and configure these fields in the TektonConfig custom resource (CR).

- **default-timeout-minutes**: This field sets the default timeout for the TaskRun and PipelineRun resources, if none is specified when creating them. If a task run or pipeline run takes more time than the set number of minutes for its execution, then the task run or pipeline run is timed out and cancelled. For example, default-timeout-minutes: 60 sets 60 minutes as default.

- **default-managed-by-label-value**: This field contains the default value given to the app.kubernetes.io/managed-by label that is applied to all TaskRun pods, if none is specified. For example, default-managed-by-label-value: tekton-pipelines.

- **default-pod-template**: This field sets the default TaskRun and PipelineRun pod templates, if none is specified.

- **default-cloud-events-sink**: This field sets the default CloudEvents sink that is used for the TaskRun and PipelineRun resources, if none is specified.

- **default-task-run-workspace-binding**: This field contains the default workspace configuration for the workspaces that a Task resource declares, but a TaskRun resource does not explicitly declare.
- **default-affinity-assistant-pod-template**: This field sets the default PipelineRun pod template that is used for affinity assistant pods, if none is specified.

- **default-max-matrix-combinations-count**: This field contains the default maximum number of combinations generated from a matrix, if none is specified.

### 3.3. CHANGING THE DEFAULT SERVICE ACCOUNT FOR OPENSIFT PIPELINES

You can change the default service account for OpenShift Pipelines by editing the `default-service-account` field in the `.spec.pipeline` and `.spec.trigger` specifications. The default service account name is `pipeline`.

**Example**

```yaml
apiVersion: operator.tekton.dev/v1alpha1
default-service-account: pipeline
default-service-account: pipeline
trigger:
default-service-account: pipeline
enable-api-fields: stable
```

### 3.4. DISABLING THE SERVICE MONITOR

You can disable the service monitor, which is part of OpenShift Pipelines, to expose the telemetry data. To disable the service monitor, set the `enableMetrics` parameter to `false` in the `.spec.pipeline` specification of the `TektonConfig` custom resource (CR):

**Example**

```yaml
enableMetrics: false
```

### 3.5. CONFIGURING PIPELINE RESOLVERS

You can configure pipeline resolvers in the `TektonConfig` custom resource (CR). You can enable or disable these pipeline resolvers:

- **enable-bundles-resolver**
- **enable-cluster-resolver**
• enable-git-resolver
• enable-hub-resolver

Example

```yaml
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  pipeline:
    enable-bundles-resolver: true
    enable-cluster-resolver: true
    enable-git-resolver: true
    enable-hub-resolver: true
```

You can also provide resolver specific configurations in the `TektonConfig` CR. For example, define the following fields in the `map[string]string` format to set configurations for each pipeline resolver:

Example

```yaml
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  pipeline:
    bundles-resolver-config:
      default-service-account: pipelines
    cluster-resolver-config:
      default-namespace: test
    git-resolver-config:
      server-url: localhost.com
    hub-resolver-config:
      default-tekton-hub-catalog: tekton
```

3.6. DISABLING CLUSTER TASKS AND PIPELINE TEMPLATES

By default, the `TektonAddon` custom resource (CR) installs `clusterTasks` and `pipelineTemplates` resources along with OpenShift Pipelines on the cluster.

You can disable installation of the `clusterTasks` and `pipelineTemplates` resources by setting the parameter value to `false` in the `.spec.addon` specification. In addition, you can disable the `communityClusterTasks` parameter.

Example

```yaml
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  addon:
```
3.7. DISABLING THE INTEGRATION OF TEKTON HUB

You can disable the integration of Tekton Hub in the web console Developer perspective by setting the `enable-devconsole-integration` parameter to `false` in the TektonConfig custom resource (CR).

Example of disabling Tekton Hub

```yaml
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  hub:
    params:
      - name: enable-devconsole-integration
        value: false
```

3.8. DISABLING THE AUTOMATIC CREATION OF RBAC RESOURCES

The default installation of the Red Hat OpenShift Pipelines Operator creates multiple role-based access control (RBAC) resources for all namespaces in the cluster, except the namespaces matching the `^(openshift|kube)-*` regular expression pattern. Among these RBAC resources, the `pipelines-scc-rolebinding` security context constraint (SCC) role binding resource is a potential security issue, because the associated `pipelines-scc` SCC has the `RunAsAny` privilege.

To disable the automatic creation of cluster-wide RBAC resources after the Red Hat OpenShift Pipelines Operator is installed, cluster administrators can set the `createRbacResource` parameter to `false` in the cluster-level TektonConfig custom resource (CR).

Example TektonConfig CR

```yaml
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  params:
    - name: createRbacResource
      value: "false"
...
3.9. AUTOMATIC PRUNING OF TASK RUNS AND PIPELINE RUNS

Stale `TaskRun` and `PipelineRun` objects and their executed instances occupy physical resources that can be used for active runs. For optimal utilization of these resources, Red Hat OpenShift Pipelines provides a pruner component that automatically removes unused objects and their instances in various namespaces.

NOTE

You can configure the pruner for your entire installation by using the `TektonConfig` custom resource and modify configuration for a namespace by using namespace annotations. However, you cannot selectively auto-prune an individual task run or pipeline run in a namespace.

3.9.1. Configuring the pruner

You can use the `TektonConfig` custom resource to configure periodic pruning of resources associated with pipeline runs and task runs.

The following example corresponds to the default configuration:

Example of the pruner configuration

```yaml
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
# ...
spec:
  pruner:
    resources:
      - taskrun
      - pipelinerun
    keep: 100
    prune-per-resource: false
    schedule: "* 8 * * *"
# ...
```

Table 3.1. Supported parameters for pruner configuration
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>schedule</strong></td>
<td>The Cron schedule for running the pruner process. The default schedule runs the process at 08:00 every day. For more information about the Cron schedule syntax, see <a href="#">Cron schedule syntax</a> in the Kubernetes documentation.</td>
</tr>
<tr>
<td><strong>resources</strong></td>
<td>The resource types to which the pruner applies. The available resource types are <code>taskrun</code> and <code>pipelinerun</code></td>
</tr>
<tr>
<td><strong>keep</strong></td>
<td>The number of most recent resources of every type to keep.</td>
</tr>
<tr>
<td><strong>prune-per-resource</strong></td>
<td>If set to <code>false</code>, the value for the <code>keep</code> parameter denotes the total number of task runs or pipeline runs. For example, if <code>keep</code> is set to 100, then the pruner keeps 100 most recent task runs and 100 most recent pipeline runs and removes all other resources. If set to <code>true</code>, the value for the <code>keep</code> parameter is calculated separately for pipeline runs referencing each pipeline and for task runs referencing each task. For example, if <code>keep</code> is set to 100, then the pruner keeps 100 most recent pipeline runs for <code>Pipeline1</code>, 100 most recent pipeline runs for <code>Pipeline2</code>, 100 most recent task runs for <code>Task1</code>, and so on, and removes all other resources.</td>
</tr>
<tr>
<td><strong>keep-since</strong></td>
<td>The maximum time for which to keep resources, in minutes. For example, to retain resources which were created not more than five days ago, set <code>keep-since</code> to 7200.</td>
</tr>
</tbody>
</table>

**NOTE**

The `keep` and `keep-since` parameters are mutually exclusive. Use only one of them in your configuration.

### 3.9.2. Annotations for automatically pruning task runs and pipeline runs

To modify the configuration for automatic pruning of task runs and pipeline runs in a namespace, you can set annotations in the namespace.

The following namespace annotations have the same meanings as the corresponding keys in the `TektonConfig` custom resource:

- `operator.tekton.dev/prune.schedule`
- `operator.tekton.dev/prune.resources`
• operator.tekton.dev/prune.keep
• operator.tekton.dev/prune.prune-per-resource
• operator.tekton.dev/prune.keep-since

NOTE
The operator.tekton.dev/prune.resources annotation accepts a comma-separated list. To prune both task runs and pipeline runs, set this annotation to "taskrun, pipelinerun".

The following additional namespace annotations are available:

• operator.tekton.dev/prune.skip: When set to true, the namespace for which the annotation is configured is not pruned.

• operator.tekton.dev/prune.strategy: Set the value of this annotation to either keep or keep-since.

For example, the following annotations retain all task runs and pipeline runs created in the last five days and delete the older resources:

Example of auto-pruning annotations

```yaml
kind: Namespace
apiVersion: v1
# ...
spec:
  annotations:
    operator.tekton.dev/prune.resources: "taskrun, pipelinerun"
    operator.tekton.dev/prune.keep-since: 7200
# ...
```

3.10. ADDITIONAL RESOURCES

• Configuring SSH authentication for Git
• Managing non-versioned and versioned cluster tasks
• Creating pipeline templates in the Administrator perspective
• Pruning objects to reclaim resources