Openshift Plugin

Bacula Systems Documentation

Contents

Contents

1 Features Summary

- OpenShift cluster resources configuration backup
- Ability to restore single OpenShift configuration resource
- Ability to restore OpenShift resource configuration to local directory
- OpenShift Persistent Volumes data backup and restore
- Ability to restore OpenShift Persistent Volumes data to local directory
- Ability to use new OpenShift CSI driver snapshot features to perform Persistent Volume data backup
- Ability to execute user defined commands on required Pod containers to prepare and clean data backup
- Configure OpenShift workload backup requirements directly with Pod annotations

2 OpenShift Backup Overview

Containers are very light system level virtualization with less overhead because programs in virtual partitions use the operating system's normal system call interface and do not need to be subjected to emulation or run in an intermediate virtual machine. OpenShift manages a set of containers to create a flexible execution environment for applications and services.

The Bacula Enterprise OpenShift plugin will save all the important Kubernetes resources which make up the application or service. This includes the following namespaced objects:

- Config Map
- Daemon Set
- Deployment
- Endpoint
- Limit Range
- Pod
- Persistent Volume Claim
- Pod Template
- Replica Set
- Replication Controller
- Resource Quota
- Secret
- Service
- Service Account
- Stateful Set
- PVC Data Archive

Note: The PVC Data is not exact Kubernetes Object but represents archive of real data existed on selected PVC.

---

and non namespaced objects:

- Namespace
- Persistent Volume
- Storage Class

All namespaced objects which belong to the particular namespace are grouped together for easy backup data browsing and recovery.

Users and service accounts can be authorized to access the server API. This process goes through authentication, authorization and admission control. To be able to successfully backup the Kubernetes resources, it is required to have a user or service account with the correct permissions and rights to be successfully authenticated and authorized to access the API server and resources to be backed up.

For resource configuration backups, the user or service account must be able to read and list the resources. In the case of PVC Data backup, it is also required that the user or service account can create and delete pods because the plugin will need to create and delete the Bacula Backup Proxy Pod during the backup.

Please see the OpenShift documentation for more details.

3 OpenShift Persistent Volume Claim Backup

All Pods in OpenShift are ephemeral and may be destroyed manually or by operations from controllers. Pods do not store data locally because stored data would be destroyed with a pod's life cycle management, so data is saved on Persistent Volumes using Persistent Volume Claim objects to control Volume Space availability.

This brings a new challenge to data backup. Fortunately most of the challenges found here are similar to standard bare metal or virtualized environments. As with bare metal and virtual machine environments, data stored in databases should be protected with dedicated Bacula Enterprise plugins that take advantage of the database backup routines.

Please refer to the appropriate Bacula Enterprise plugin whitepapers for more details on database backups.

On the other hand, most non-database applications store data as simple flat files we can backup as-is without forcing complicated transactions or data consistency procedures. This use case is handled directly with the OpenShift plugin using a dedicated Bacula Backup Proxy Pod executed in the cluster.

If the container application is more complex, it is possible to execute commands inside the container to quiesce the application.

- before the snapshot
- after the snapshot
- after the backup of the container

A problem with command execution can abort the backup of the container with the **run.*.failonerror** annotation. You can find detailed description of this feature at *CSI Snapshot Support*.

A Bacula Backup Proxy Pod is a service executed automatically by the OpenShift plugin which manages secure access to Persistent Volume data for the plugin. It is executed on the OpenShift cluster infrastructure and requires a network connection to the OpenShift plugin for data exchange on backup and restore operations. No external cluster service like `NodePort`, `LoadBalancer`, `Ingress` or `Host Based Networking` configuration is required to use this feature.

It is also not required to permanently deploy and run this service on the cluster itself as it is executed on demand. The Bacula Backup Proxy Pod does not consume any valuable compute resources outside of the backup window. You can even operate your OpenShift backup solution (Bacula Enterprise service with OpenShift plugin) directly from your on-premise backup infrastructure to backup a public Kubernetes cluster (it requires a simple port forwarding firewall rule) or use public backup infrastructure to backup on-premise Kubernetes cluster(s). Support for these varied architecture modes is built into the OpenShift plugin. It is designed to be a *One-Click* solution for OpenShift backups.

Starting from version `1.1.0` of the OpenShift plugin, you can backup and restore any PVC data including PVCs not attached to any running Kubernetes Pod. This removes a previous limitation in this area.

4 CSI Snapshot Support

Starting from Bacula Enterprise version 12.6 and OpenShift plugin version 2.0, OpenShift CSI Snapshot functionality support together with a bunch of other features was added. Starting from this version you can use CSI Snapshots to acquire a consistent data view of selected Volume. Additionally, you can configure remote command execution on a selected Container of the Pod. You can configure command execution just before or after a Pod backup and just after snapshot creation.

Our plugin uses the volume clone api when doing a volume snapshot. CSI drivers may or may not have implemented the volume cloning functionality.

5 OpenShift Pod Annotations

The **CSI Snapshot Support** feature described in *CSI Snapshot Support* comes with a configuration of Volume data backup using Pod annotations. This feature allows you to define what volumes (PVC Data) to backup, where and what commands to execute, and how to react to some failures to achieve the best results from data snapshot functionality. You can select which volumes mounted at the Pod you want to backup, the preferred backup mode for the Pod, and what commands you want to execute.

The supported annotations are:

bacula/backup.mode:[snapshot|standard] defines how to handle PVC Data backups. If not defined, the default is **snapshot**.

bacula/backup.volumes:<pvcname[,pvcname2...]> defines what volumes will be backed up from this Pod. Multiple PVC names may be selected as a comma separated list. This annotation is required if you want to backup volumes on this Pod. Any other annotations are optional.

bacula/run.before.job.container.command:<container>/<command> defines what command to execute on which container of the Pod before any snapshot and data backup of this Pod occurs. An asterisk (*) as <container> name means to execute <command> on all containers.

bacula/run.before.job.failjobonerror:[yes|no] defines whether or not to fail the job when the exit code of the executed command is not zero. The default is yes.

bacula/run.after.job.container.command:<container>/<command> defines what command to execute on which container of the Pod after all snapshot and data backup of this Pod. An asterisk (*) as <container> name means to execute <command> on all containers.

bacula/run.after.job.failjobonerror:[yes|no] defines whether or not to fail the job when exit code of the executed command is not zero. The default is no.

bacula/run.after.snapshot.container.command:<container>/<command> defines what command to execute on which container of the Pod after all snapshot creations and just before any data backup. An asterisk (*) as <container> name means to execute <command> on all containers.

bacula/run.after.snapshot.failjobonerror:[yes|no] defines to fail the job when exit code of the executed command is not zero. The default is no.

Pod annotations is an extension to the current PVC Data backup feature available with the **pvcdata=...** plugin parameter as described in *Estimate and Backup Plugin Parameters*. This is an independent function which may be used together with the functionality described above, especially since both use the same data archive stream handling with Bacula Backup Pod.

All you need to use a new feature is to configure selected Pod annotations and make sure that the backup for a required OpenShift namespace is properly configured. There is no need for any plugin configuration modifications. A Pod's volumes will be backed up automatically.

5.1 Examples

Below you can find some examples how to configure Bacula annotations in OpenShift Pods.

In the example below you will use a simple Linux command `sync` to synchronize cached writes to persistent storage before volume snapshot.

```
apiVersion: v1
  kind: Pod
  metadata:
    name: app1
    namespace: default
    annotations:
      bacula/backup.mode: snapshot
      bacula/run.before.job.container.command: "*/sync -f /data1; sync -f /data2"
      bacula/run.before.job.failjobonerror: "no"
      bacula/backup.volumes: "pvc1,  pvc2"
  spec:
    containers:
    - image: ubuntu:latest
      name: test-container
      volumeMounts:
      - name: pvc1
        mountPath: /data1
      - name: pvc2
        mountPath: /data2
      volumes:
      - name: pvc1
        persistentVolumeClaim:
          claimName: pvc1
      - name: pvc2
        persistentVolumeClaim:
          claimName: pvc2
```

In the example below you will use PostgreSQL's database data files quiesce feature to perform consistent backup with snapshot.

Note: The final PostgreSQL backup solution requires more configuration and preparation which was skipped in this example to make it clear

---

The first command (*run.before.job.container.command*) freezes writes to database files and the second (*run.after.snapshot.container.command*) resumes standard database operation as soon as PVC snapshot becomes ready.

```
apiVersion: v1
kind: Pod
metadata:
  name: postgresql13
```

(continues on next page)

(continued from previous page)

```
  namespace: default
  annotations:
    bacula/backup.mode: standard
    bacula/run.before.job.container.command: "*//bin/startpgsqlbackup.sh"
    bacula/run.after.snapshot.container.command: "*//bin/stoppgsqlbackup.sh"
    bacula/run.after.snapshot.failjobonerror: "yes"
    bacula/backup.volumes: "pgsql"
  spec:
    containers:
    - image: postgresql:13
      name: postgresql-server
      volumeMounts:
      - name: pgsql
        mountPath: /var/lib/pgsql
      volumes:
      - name: pgsql
        persistentVolumeClaim:
          claimName: pgsql-volume
```

5.2 Run Container Command annotation

All flavors of the Run Container Command parameters are remotely executed using the OpenShift Pod remote execution API. Every command is prepared to execute with a standard Linux shell `/bin/sh`. This requires that a container image has to have the specified shell available. Using command shell execution gives flexibility to command execution or even allows for preparation of small scripts without additional container image customization.

6 Installation

Please create the following bacula-fd.yaml pod configuration file to be used for deployment:

```
apiVersion: v1
kind: Pod
metadata:
  name: bacula-fd
  labels:
    app: bacula-fd
spec:
  containers:
  - name: web
    image: registry.connect.redhat.com/bacula-enterprise/bacula-enterprise-openshift-
↪plugin-1260
    env:
      - name: MASTER
        value: "true"
    volumeMounts:
    - name: bacula-conf
      mountPath: /opt/bacula/etc
  volumes:
    - name: bacula-conf
```

(continues on next page)

(continued from previous page)

```
      configMap:
        name: bacula-configmap
```

Once you have the pod configuration file, please use the following procedure to deploy it:

Login to the OpenShift cluster using an account/service account with enough permissions to deploy the pod:

```
oc login -u kubeadmin -p <password> <cluster URI>
```

Create a bacula-fd.conf file that will be used to create the configmap resource for the Bacula File Daemon pod. The bacula-fd.conf file must be in your OpenShift cluster file system. For example:

```
Director {
    Name = "bacula-dir"
    Password = "bacula-dir-fd-password"
}
FileDaemon {
    Name = "openshift-fd"
    Description = "OpenShift Plugin File Daemon"
    PidDirectory = "/opt/bacula/working"
    PluginDirectory = "/opt/bacula/plugins"
    WorkingDirectory = "/opt/bacula/working"
}
Messages {
    Name = "Standard"
    Director  = "bacula-dir" = All,!Skipped,!Restored
}
```

Please modify the above example with the Director name and password to be used in your Bacula Enterprise environment.

Then create the configmap resource:

```
oc create configmap <configmapname> --from-file=/path/to/bacula-fd.conf --from-file=/
↪path/to/cluster/kubeconfig
```

And create the bacula-fd pod:

```
oc apply -f ./bacula-fd.yaml
```

7 Bacula Backup Proxy Pod Image

For OpenShift PVC Data backup functionality, the Bacula Enterprise **OpenShift** Plugin requires a dedicated Bacula Backup Proxy Pod which is automatically deployed using an image that is available in the `bacula-enterprise-openshift-tools` package.

This image should be installed manually on your local Docker images registry service which is available on your OpenShift cluster as a source for application images.

Installation of the image can be performed with the following example commands:

```
# cd /opt/bacula/lib
# docker load -i bacula-backup-<timestamp>.tar
```

(continues on next page)

(continued from previous page)

```
# docker image tag bacula-backup:<timestamp> <registry>/bacula-backup:<timestamp>
# docker push <registry>/bacula-backup:<timestamp>
```

Where <timestamp> is the image version shipped with the above package and <registry> is the location of your Docker images registry service. The exact procedure depends on your Kubernetes cluster deployment, so please verify the above steps before attempting to run the docker commands.

You can use any registry service available for your cluster, public or private, i.e. gcr.io/.

Depending on your cluster configuration it may be necessary to set the **baculaimage=<name>** plugin parameter (see section *Backup and Restore Plugin Parameters* for details) to define which repository and container image to use. The default for this parameter is bacula-backup:<timestamp> which may not be correct for your deployment.

Another example where you will need to modify the Bacula Backup Proxy Pod Image is in the case where your registry requires authentication. Please see the section *Advanced Bacula Backup Proxy Pod Deployment* for more details.

8 Backup and Restore Operations

9 Backup

The plugin can backup a number of Kubernetes Resources including: Deployments, Pods, Services or Persistent Volume Claims, check chapter *OpenShift Backup Overview* for a complete list.

The backup will create a single (.yaml) file for any Kubernetes Resource which is saved. For PVC Data backup functionality the OpenShift plugin generates a data archive as a single <pvcname>.tar archive file. The resources are organized inside the Bacula catalog to facilitate browsing and restore operations. In the Bacula catalog, Kubernetes resources are represented as follows:

- /@openshift/namespaces/<namespace>.yaml - Namespace definition
- /@openshift/namespaces/<namespace>/<resource>/<name>.yaml - Resource definitions in the namespace
- /@openshift/namespaces/<namespace>/persistentvolumeclaim/<pvcname>.tar - PVC Data backup in the selected namespace
- /@openshift/persistentvolumes/<pvname>.yaml - Persistent Volume definition
- /@openshift/storageclass/<scname>.yaml - Storage Class definition

All supported Kubernetes Resources will be saved if no filter options are set. You may limit which resources are saved using filtering options described in chapter *Estimate and Backup Plugin Parameters*. By default, if no filter options are set, all supported Kubernetes Resources will be saved. To see the Kubernetes Resources that may be filtered, a listing mode is available. This mode is described in chapter *Resource listing*.

10 Restore

The OpenShift plugin provides two targets for restore operations:

- Restore to an OpenShift cluster
- Restore to a local directory

10.1 Restore to an OpenShift Cluster

To use this restore method, the **where=/** parameter of a Bacula `restore` command is used. You can select any supported Kubernetes Resource to restore, or batch restore the whole namespace or even multiple namespaces. If you select a single resource to restore it will be restored as is without any dependent objects. In most cases, for (`Config Maps`, `Secrets`, `Services`, etc.) this is fine and restore will always succeed. On the other hand, compound objects (`Pods`, `Deployments`, etc.) won't be ready unless all dependencies are resolved during the restore. In this case you should make sure that you select all required resources to restore.

In OpenShift, a successful resource restore doesn't necessarily result in the service successfully coming online. In some cases further monitoring and investigation will be required. For example:

- *Container image is unavailable*.
- *Volume Claim cannot provision new volume*.
- *Untrusted Image Repository*.
- *Infrastructure limits exceeded, i.e. a number of Pods or Memory allocations*.
- *etc...*

All example cases above must be resolved by the OpenShift administrator. When all issues are resolved, the resource should automatically come online. If not, it may be necessary to repeat a restore to redeploy the Resource configuration.

The OpenShift plugin does not wait for a Resource to come online during restore. It checks the Resource creation or replace operation status and reports any errors in the job log. The only exception to this is PVC Data restore, when the OpenShift plugin will wait for a successful archive data restore. This operation is always executed at the end of the namespace recovery (when pvcdata is restored with other K8S objects) and should wait for proper PVC mount.

10.2 Restore to a Local Directory

To use this mode, the **where=/some/path** Bacula `restore` parameter is set to a full path on a server where the Bacula File Daemon and OpenShift plugin are installed. If the path does not exist, it will be created by the OpenShift plugin. With this restore mode you can restore any saved Kubernetes Resource including PVC Data archive file to a location on disk.

Please note that the OpenShift Client may not have enough space or sufficient permissions to restore in local directories of the bacula-fd pod. Thus It may be required to provide a persistent volume to the bacula-fd pod to perform the restore to a local directory.

11 Plugin Configuration

The plugin is configured using **Plugin Parameters** defined in a **FileSets** -> **Include** section of the Bacula Enterprise Director configuration.

12 Generic Plugin Parameters

The following OpenShift plugin parameters affect any type of Job (Backup, Estimate, or Restore).

abort_on_error[=<0 or 1>] specifies whether or not the plugin should abort execution (and the Bacula Job) if a fatal error occurs during a Backup, Estimate, or Restore operation. If not specified, the default value is 0.

This parameter is optional.

config=</path/to/file> points to the location of the config file which defines how to connect to the OpenShift cluster. Please check the OpenShift documentation for more details about the *kubeconfig* file. If this directive is omitted and no other access method is configured then a default config file location will be used - `$HOME/.kube/config`.

This parameter is optional.

incluster if this directive is defined then a standard in-cluster access to the OpenShift API will be used. This option should be used only when the Bacula File Daemon and OpenShift plugin are deployed as an OpenShift cluster service in a Pod. In this case, OpenShift itself will provide the required access credentials. This option won't work when executed outside of an OpenShift cluster and services.

This parameter is optional.

host=<url-openshift-api> defines an OpenShift API url. This option is used only with **token** parameter described below. If this option is specified, then both parameters are required.

This parameter is optional.

token=<bearer-token> defines a `Bearertoken` used for authorization to the OpenShift API available at **host=<url-openshift-api>**. This option is used only with **host** parameter described above. You can read more about this type of authentication at: https://swagger.io/docs/specification/authentication/bearer-authentication/

This parameter is optional.

verify_ssl[=<0 or 1>] specifies whether or not the plugin should verify an OpenShift API certificate when the connection uses SSL/TLS. If set to **verify_ssl=0** then verification will be disabled. This is useful when connecting to an OpenShift API server with a self-signed certificate. The default behavior if this parameter if not set is to perform proper certificate validation.

This parameter is optional.

ssl_ca_cert=</path/to/file> specifies a file with the CA certificate used to customize the OpenShift API server identity to verify the peer.

This parameter is optional.

timeout=<seconds> specifies the number of seconds for various network operations. Examples include: waiting for Bacula Backup Proxy Pod connection or OpenShift API operations, waiting for Pod execution or removal. The default is 600 seconds. The minimum timeout you may set is 1 second. When an invalid value is set the default will be used.

This parameter is optional.

debug[=1] specifies that the plugin backend will generate an execution debug file at location `/bacula/working/backendplugin/`. This file can help with troubleshooting the job execution if something goes wrong. If not defined then no debug file will be generated.

This parameter is optional.

13 Estimate and Backup Plugin Parameters

These plugin parameters are relevant only for Backup and Estimate jobs:

namespace=<name> specifies an OpenShift namespace name which you want to backup. Multiple `namespace=<name>` parameters are allowed if you want to backup multiple namespaces. If a namespace with `name` can not be found its backup will be silently ignored. If this parameter is not set, all namespaces will be saved.

This parameter is optional.

persistentvolume=<name> specifies an OpenShift persistent volume configuration you want to backup. Multiple `persistentvolume=<name>` parameters are allowed if you want to backup multiple volumes. You can use standard shell wildcard pattern matching to select multiple volumes using a single `persistentvolume` parameter. If a persistent volume with `name` can not be found its backup will be silently ignored. If this parameter is not set, all persistent volume configurations will be saved unless **pvconfig=0** is set as described below.

This parameter is optional.

pvconfig=[0|1] this option is used to disable persistent volume configuration backups when **pvconfig=0** is set. The default is to backup persistent volume configurations.

This parameter is optional.

storageclass=<name> specifies an OpenShift Storage Class configuration to be backed up. Multiple `storageclass=<name>` parameters are allowed if you want to backup multiple resources. You can use standard shell wildcard pattern matching to select multiple volumes using a single `storageclass` parameter. If a storage class with `name` can not be found, its backup will be silently ignored. If this parameter is not set, all storage class configuration will be saved unless **scconfig=0** is set as described below.

This parameter is optional.

scconfig=[0|1] this option is used to disable Storage Class configuration backups when **scconfig=0** is set. The default is to backup Storage Class resource configurations.

This parameter is optional.

pvcdata[=<pvcname>] specifies an OpenShift Persistent Volume Claim name you want to make a PVC Data archive for. As all PVCs are namespaced objects, to use this option you should select a required namespace with **namespace=<name>** parameter. If you define a simple **pvcdata** parameter without the equals sign ("=") and subsequent value, all detected persistent volume claims will be selected for the PVC Data archive backup.

This parameter is optional.

If none of the parameters above are specified, then all available Namespaces and Persistent Volume Configurations will be backed up. However, the plugin does not force a PVC Data archive backup in this case.

14 Backup and Restore Plugin Parameters

These plugin parameters are relevant only for Backup and Restore jobs when the PVC Data archive functionality is used:

fdaddress=<IP or name> is the IP address or host name where the OpenShift plugin should listen for incoming connections from a Bacula Backup Proxy Pod. This parameter, combined with **fdport=<number>** below will define a socket to listen on. The default is to listen on all available interfaces (`0.0.0.0`) which should be fine in most cases.

This parameter is optional if **pluginhost=<IP or name>** is defined.

fdport=<number> is a port number on which OpenShift plugin should listen for incoming connections from a Bacula Backup Proxy Pod. This parameter, combined with **fdaddress=<IP or name>** above will define a socket to listen on. The default is to listen on port `9104`.

This parameter is optional.

pluginhost=<IP or name> is the entry point address where a Bacula Backup Proxy Pod should connect during backup or restore operations. The name should be resolvable on the OpenShift cluster, otherwise an IP address must be used here. This with **pluginport=<number>** parameter below will define the exact service entry point. The default is to use the value from **fdaddress=<IP or name>** parameter above.

This parameter is required when **fdaddress=<IP or name>** is not defined. Otherwise it is optional.

pluginport=<number> is the port number for service entry point address where the Bacula Backup Proxy Pod should connect during backup or restore operations. This, combined with the **pluginhost=<IP or name>** parameter above define the exact service entry point. The default is to use the value from the **fdaddress=<IP or name>** parameter above. When neither is defined the default `9104` port number will be used.

This parameter is optional.

fdcertfile=<path> is the SSL certificate file location for the OpenShift plugin endpoint service data connection for a Bacula Backup Proxy Pod. The certificate and key (see **fdkeyfile=<path>** below) files are required for proper Bacula Backup Proxy Pod. You can create and use any valid SSL certificate including a self-signed one. You can even just use the certificate generated during the OpenShift plugin installation located at `/opt/bacula/etc/bacula-backup.cert` The default is to use the certificate generated during plugin installation.

This parameter is optional.

fdkeyfile=<path> is an SSL private key file location for the SSL certificate defined by **fdcertfile=<path>** above. An unencrypted private key must be used for this purpose. The default is to use a private key file created during plugin installation at `/opt/bacula/etc/bacula-backup.key` when **fdcertfile=<path>** above is not defined (the default). Otherwise the plugin will refer to the same certificate file from **fdcertfile=<path>** and use it as a `.pem` Container.

This parameter is optional.

baculaimage=<name> is a Bacula Backup Proxy Pod Container image registry location for your cluster as described in the image installation chapter *Bacula Backup Proxy Pod Image*. In most cases this parameter will consist of your Docker images registry location with the tag of the required image. The default for this parameter is `bacula-backup:<timestamp>` and may not match your cluster configuration.

This parameter is optional.

imagepullpolicy=<Always|Never|ifNotPresent> sets the OpenShift pod image pull policy during Bacula Backup Proxy Pod Container deployment. You can configure OpenShift to **Always** or **Never** pull the image from the repository, or pull it only when not available with **ifNotPresent**. The option value is case insensitive. If this parameter is not defined, the default **ifNotPresent** will be used.

This parameter is optional.

15 Restore Plugin Parameters

During restore, the OpenShift plugin will use the same parameters which were set for the backup job and saved in the catalog. During restore, you may change any of the parameters described in chapter *Generic Plugin Parameters* and *Backup and Restore Plugin Parameters*. In addition to the options used for backups, the **outputformat** option can be used during restore. This option specifies the file format when restoring to a local filesystem. You can choose the restore output in `JSON` or `YAML`. If not defined the restored files will be saved in `YAML`.

outputformat: <JSON or YAML> specifies the file format when restoring to a local filesystem as described above.

This parameter is optional.

16 FileSet Examples

In the example below, all OpenShift Namespaces, Resources and Persistent Volume Configurations will be backed up using the default OpenShift API access credentials.

```
FileSet {
  Name = FS_OpenShift_All
  Include {
    Plugin = "openshift:"
  }
}
```

In this example, we will backup a single Namespace using the kube config file authentiaton method.

```
FileSet {
  Name = FS_OpenShift_default_namespace
  Include {
    Plugin = "openshift: namespace=default config=/opt/bacula/etc/kube_cofig"
  }
}
```

The same example as above, but with a Persistent Volume:

```
FileSet {
  Name = FS_OpenShift_default_namespace
  Include {
    Plugin = "openshift: namespace=default config=/opt/bacula/etc/kube_cofig \
              persistentvolume=myvol"
  }
}
```

This example backs up a single Namespace and all detected PVCs in this Namespace using a defined listening and entry point address and the default connection port:

```
FileSet {
  Name = FS_OpenShift_test_namespace
  Include {
    Plugin = "openshift: namespace=test pvcdata fdaddress=10.0.10.10"
  }
}
```


The same example as above, but using different listening and entry point addresses as may be found when the service is behind a firewall using port forwarding features:

```
FileSet {
  Name = FS_OpenShift_test_namespace_through_firewall
  Include {
    Plugin = "openshift: namespace=test pvcdata=plugin-storage fdaddress=10.0.10.10 \
        pluginhost=backup.example.com pluginport=8080"
  }
}
```

The configuration above is designed for use in situations where the Bacula server components are located on-premise and behind a firewall with no external ports allowed in, but must back up data on an external OpenShift cluster.

17 Restore examples

18 Restore to OpenShift Cluster

To restore Kubernetes resources to an OpenShift cluster, the administrator should execute the restore command and specify the **where** parameter as in this example:

```
* restore where=/
```

and then set any other required restore plugin parameters for the restore.

```
* restore where=/
...
$ cd /@openshift/namespaces/plugintest/configmaps/
cwd is: /@openshift/namespaces/plugintest/configmaps/
$ ls
plugintest-configmap.yaml
$ add *
1 file marked.
$ done
Bootstrap records written to /opt/bacula/working/bacula-dir.restore.1.bsr

The Job will require the following (*=>InChanger):
Volume(s)                 Storage(s)                SD Device(s)
===========================================================================

Vol005                    File1                     FileChgr1

Volumes marked with "*" are in the Autochanger.


1 file selected to be restored.

Run Restore job
JobName:         RestoreFiles
Bootstrap:       /opt/bacula/working/bacula-dir.restore.1.bsr
Where:           /
```

(continues on next page)

(continued from previous page)

```
Replace:         Always
FileSet:         Full Set
Backup Client:   bacula-fd
Restore Client:  bacula-fd
Storage:         File1
When:            2019-09-30 12:39:13
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (yes/mod/no): mod
Parameters to modify:
    1: Level
    2: Storage
    3: Job
    4: FileSet
    5: Restore Client
    6: When
    7: Priority
    8: Bootstrap
    9: Where
    10: File Relocation
    11: Replace
    12: JobId
    13: Plugin Options
Select parameter to modify (1-13): 13
Automatically selected : openshift: config=/home/test_user/.kube/config
Plugin Restore Options
config:              radekk/.kube/config  (*None*)
host:                *None*               (*None*)
token:               *None*               (*None*)
username:            *None*               (*None*)
password:            *None*               (*None*)
verify_ssl:          *None*               (True)
ssl_ca_cert:         *None*               (*None*)
outputformat:        *None*               (RAW)
Use above plugin configuration? (yes/mod/no): mod
You have the following choices:
    1: config (K8S config file)
    2: host (K8S API server URL/Host)
    3: token (K8S Bearertoken)
    4: verify_ssl (K8S API server cert verification)
    5: ssl_ca_cert (Custom CA Certs file to use)
    6: outputformat (Output format when saving to file (JSON, YAML))
    7: fdaddress (The address for listen to incoming backup pod data)
    8: fdport (The port for opening socket for listen)
    9: pluginhost (The endpoint address for backup pod to connect)
    10: pluginport (The endpoint port to connect)
Select parameter to modify (1-8): 1
Please enter a value for config: /root/.kube/config
Plugin Restore Options
config:              /root/.kube/config   (*None*)
host:                *None*               (*None*)
```

(continues on next page)

(continued from previous page)

```
token:               *None*               (*None*)
verify_ssl:          *None*               (True)
ssl_ca_cert:         *None*               (*None*)
outputformat:        *None*               (RAW)
fdaddress:           *None*               (*FDAddress*)
fdport:              *None*               (9104)
pluginhost:          *None*               (*FDAddress*)
pluginport:          *None*               (9104)
Use above plugin configuration? (yes/mod/no): yes
Job queued. JobId=1084
```

The plugin does not wait for Kubernetes Resources to become ready and online in the same way as the `kubectl` or the `oc` commands.

19 Restore to a Local Directory

It is possible to restore any Kubernetes Resource(s) to file without loading them into a cluster. To do so, the **where** restore option should point to the local directory:

```
* restore where=/tmp/bacula/restores
...
$ cd /@openshift/namespaces/
cwd is: /@openshift/namespaces/
$ ls
bacula/
cattle-system/
default/
graphite/
ingress/
plugintest/
$ add plugintest
25 files marked.
$ done
Bootstrap records written to /opt/bacula/working/bacula-dir.restore.2.bsr

The Job will require the following (*=>InChanger):
Volume(s)                 Storage(s)                SD Device(s)
===========================================================================

Vol005                    File1                     FileChgr1

Volumes marked with "*" are in the Autochanger.


25 files selected to be restored.

Run Restore job
JobName:         RestoreFiles
Bootstrap:       /opt/bacula/working/bacula-dir.restore.2.bsr
Where:           /tmp/bacula/restores
```

(continues on next page)

(continued from previous page)

```
Replace:         Always
FileSet:         Full Set
Backup Client:   bacula-fd
Restore Client:  bacula-fd
Storage:         File1
When:            2019-09-30 12:58:16
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (yes/mod/no): mod
Parameters to modify:
    1: Level
    2: Storage
    3: Job
    4: FileSet
    5: Restore Client
    6: When
    7: Priority
    8: Bootstrap
    9: Where
    10: File Relocation
    11: Replace
    12: JobId
    13: Plugin Options
Select parameter to modify (1-13): 13
Automatically selected : openshift: config=/home/radekk/.kube/config debug=1
Plugin Restore Options
config:              *None*              (*None*)
host:                *None*              (*None*)
token:               *None*              (*None*)
verify_ssl:          *None*              (True)
ssl_ca_cert:         *None*              (*None*)
outputformat:        *None*              (RAW)
fdaddress:           *None*              (*FDAddress*)
fdport:              *None*              (9104)
pluginhost:          *None*              (*FDAddress*)
pluginport:          *None*              (9104)
Use above plugin configuration? (yes/mod/no): mod
You have the following choices:
    1: config (K8S config file)
    2: host (K8S API server URL/Host)
    3: token (K8S Bearertoken)
    4: verify_ssl (K8S API server cert verification)
    5: ssl_ca_cert (Custom CA Certs file to use)
    6: outputformat (Output format when saving to file (JSON, YAML))
    7: fdaddress (The address for listen to incoming backup pod data)
    8: fdport (The port for opening socket for listen)
    9: pluginhost (The endpoint address for backup pod to connect)
    10: pluginport (The endpoint port to connect)
Select parameter to modify (1-8): 8
Please enter a value for outputformat: JSON
Plugin Restore Options
```

(continues on next page)

(continued from previous page)

```
config:              *None*              (*None*)
host:                *None*              (*None*)
token:               *None*              (*None*)
verify_ssl:          *None*              (True)
ssl_ca_cert:         *None*              (*None*)
outputformat:        *None*              (RAW)
fdaddress:           *None*              (*FDAddress*)
fdport:              *None*              (9104)
pluginhost:          *None*              (*FDAddress*)
pluginport:          JSON                (9104)
Use above plugin configuration? (yes/mod/no): yes
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /opt/bacula/working/bacula-dir.restore.2.bsr
Where:           /tmp/bacula/restores
Replace:         Always
FileSet:         Full Set
Backup Client:   bacula-fd
Restore Client:  bacula-fd
Storage:         File1
When:            2019-09-30 12:58:16
Catalog:         MyCatalog
Priority:        10
Plugin Options:  User specified
OK to run? (yes/mod/no):
Job queued. JobId=1085
```

Output format conversion at restore time will format all data in a human readable format. You can find an example of this restore below.

```
# cat /tmp/bacula/restores/namespaces/plugintest/plugintest.json
{
    "apiVersion": "v1",
    "kind": "Namespace",
    "metadata": {
        "annotations": {
            "field.cattle.io/projectId": "c-hb9ls:p-bm6cw",
            "lifecycle.cattle.io/create.namespace-auth": "true"
        },
        "cluster_name": null,
        "creation_timestamp": "2019-09-25T16:31:03",
        "deletion_grace_period_seconds": null,
        "deletion_timestamp": null,
        "finalizers": [
        "controller.cattle.io/namespace-auth"
        ],
        "generate_name": null,
        "generation": null,
        "initializers": null,
        "labels": {
            "field.cattle.io/projectId": "p-bm6cw"
        },
```

(continues on next page)

(continued from previous page)

```
        "name": "plugintest",
        "namespace": null,
        "owner_references": null,
        "resource_version": "11622",
        "self_link": "/api/v1/namespaces/plugintest",
        "uid": "dd873930-dfb1-11e9-aad0-022014368e80"
    },
    "spec": {
        "finalizers": [
        "kubernetes"
        ]
    },
    "status": {
        "phase": "Active"
    }
}
```

The supported output transformations are: `JSON` and `YAML`.

20 Restore PVC Data Archive

Here we describe functionalities and requirements related to pvcdata restore.

20.1 Local Directory Restore

The procedure to restore a PVC Data archive file to a local directory is basically the same as restoring the Kubernetes Resource configuration file as described in *Restore to a Local Directory*. However, output transformation is unavailable and ignored when restoring PVC data. Restore of this data will create a `tar` archive file you can manually inspect and use.

20.2 Restore to PVC

This procedure is similar to the one described in PVC Data backup and uses the same Bacula Backup Proxy Pod image. During restore, the plugin uses the same endpoint configuration parameters so it is not necessary to setup it again. If your endpoint parameters have changed you can update them using Bacula plugin restore options modification as in example below:

```
*restore select all done where=/
(...)
OK to run? (yes/mod/no): mod
Parameters to modify:
    1: Level
    2: Storage
    3: Job
    4: FileSet
    5: Restore Client
    6: When
    7: Priority
    8: Bootstrap
```

(continues on next page)

(continued from previous page)

```
    9: Where
    10: File Relocation
    11: Replace
    12: JobId
    13: Plugin Options
Select parameter to modify (1-13): 13
Automatically selected : openshift: namespace=plugintest pvcdata pluginhost=example.com
Plugin Restore Options
config:              *None*              (*None*)
host:                *None*              (*None*)
token:               *None*              (*None*)
verify_ssl:          *None*              (True)
ssl_ca_cert:         *None*              (*None*)
outputformat:        *None*              (RAW)
fdaddress:           *None*              (*FDAddress*)
fdport:              *None*              (9104)
pluginhost:          *None*              (*FDAddress*)
pluginport:          *None*              (9104)
Use above plugin configuration? (yes/mod/no): mod
You have the following choices:
    1: config (K8S config file)
    2: host (K8S API server URL/Host)
    3: token (K8S Bearertoken)
    4: verify_ssl (K8S API server cert verification)
    5: ssl_ca_cert (Custom CA Certs file to use)
    6: outputformat (Output format when saving to file (JSON, YAML))
    7: fdaddress (The address for listen to incoming backup pod data)
    8: fdport (The port for opening socket for listen)
    9: pluginhost (The endpoint address for backup pod to connect)
    10: pluginport (The endpoint port to connect)
Select parameter to modify (1-10): 9
Please enter a value for pluginhost: newbackup.example.com
Plugin Restore Options
config:              *None*              (*None*)
host:                *None*              (*None*)
token:               *None*              (*None*)
verify_ssl:          *None*              (True)
ssl_ca_cert:         *None*              (*None*)
outputformat:        *None*              (RAW)
fdaddress:           *None*              (*FDAddress*)
fdport:              *None*              (9104)
pluginhost:          newbackup.example.com (*FDAddress*)
pluginport:          *None*              (9104)
Use above plugin configuration? (yes/mod/no): yes
```

You can restore all data available from the backup archive for a selected Persistent Volume Claim and all data will be overwritten, ignoring the `Replace` job parameter. Please take note of this behavior, which may change in the future.

21 Other

22 Resource listing

The Bacula Enterprise OpenShift plugin supports the "plugin listing" feature of Bacula Enterprise 8.x or newer. This mode allows the plugin to display some useful information about available Kubernetes resources such as:

- List of OpenShift namespaces
- List of OpenShift Persistent Volumes
- List of OpenShift storage class resources

The feature uses the special **.ls** command with a **plugin=<plugin>** parameter.

The command requires the following parameters to be set:

client=<client> A Bacula Client name with the OpenShift plugin installed.

plugin=<plugin> A plugin name, which would be **openshift:** in this case, with optional plugin parameters as described in section *Generic Plugin Parameters*.

path=<path> An object path to display.

The supported values for a **path=<path>** parameter are:

/ to display Object types available to list

namespaces to display a list of Namespaces

persistentvolumes to display a list of Persistent Volumes

storageclass to display a list of Storage Class Resources

namespaces/<name>/pvcdata to display all available Persistent Volume Claims available in Namespace <name>

To display available Object types, follow the following command example:

```
*.ls plugin=openshift: client=openshift-fd path=/
Connecting to Client openshift-fd at localhost:9102
drwxr-x---   1 root     root 2018-09-28 14:32:20  /namespaces
drwxr-x---   1 root     root 2018-09-28 14:32:20  /persistentvolumes
drwxr-x---   1 root     root 2018-09-28 14:32:20  /storageclass
2000 OK estimate files=2 bytes=0
```

To display the list of all available Kubernetes namespaces, the following command example can be used:

```
*.ls plugin=openshift: client=openshift-fd path=namespaces
Connecting to Client openshift-fd at localhost:9102
drwxr-xr-x   1 root     root 2019-09-25 16:39:56  /namespaces/default
drwxr-xr-x   1 root     root 2019-09-25 16:39:56  /namespaces/kube-public
drwxr-xr-x   1 root     root 2019-09-25 16:39:56  /namespaces/kube-system
drwxr-xr-x   1 root     root 2019-09-25 16:46:19  /namespaces/cattle-system
drwxr-xr-x   1 root     root 2019-09-27 13:04:01  /namespaces/plugintest
2000 OK estimate files=5 bytes=0
```

To display the list of available Persistent Volume Claims which could be used for PVC Data archive feature selection, you can use the following example command for the `mysql` namespace:


```
*.ls client=openshift-fd plugin="openshift:" path=/namespaces/mysql/pvcdata
Connecting to Client openshift-fd at openshift:9102
-rw-r-----   1 root     root 2019-10-16 14:29:38  /namespaces/mysql/pvcdata/mysql-mysql
2000 OK estimate files=1 bytes=0
```

To display the list of all available Persistent Volumes, the following command example can be used:

```
*.ls plugin=openshift: client=openshift-fd path=persistentvolumes
Connecting to Client openshift-fd at localhost:9102
-rw-r-----  1073741824 2019-09-25 /persistentvolumes/pvc-bfaebd0d-dfad-11e9-a2cc-
↪42010a8e0174
-rw-r-----  1073741824 2019-09-25 /persistentvolumes/pvc-b1a49497-dfad-11e9-a2cc-
↪42010a8e0174
-rw-r-----  1073741824 2019-09-25 /persistentvolumes/pvc-949cb638-dfad-11e9-a2cc-
↪42010a8e0174
-rw-r-----  1073741824 2019-09-25 /persistentvolumes/pvc-9313388c-dfad-11e9-a2cc-
↪42010a8e0174
-rw-r----- 10737418240 2019-09-24 /persistentvolumes/myvolume
2000 OK estimate files=5 bytes=15,032,385,536
```

The volume lists display a Volume Storage size which does not reflect the actual configuration size during backup.

To display the list of all defined Storage Class Resources, the following command example can be used:

```
*.ls plugin=openshift: client=openshift-fd path=storageclass
Connecting to Client openshift-fd at openshift:9102
-rw-r----- 1024 2020-07-27 13:39:48  /storageclass/local-storage
-rw-r----- 1024 2020-07-23 16:14:13  /storageclass/default-postgresql-1
-rw-r----- 1024 2020-07-24 11:47:02  /storageclass/local-storage-default
-rw-r----- 1024 2020-07-23 12:00:02  /storageclass/standard
2000 OK estimate files=4 bytes=4,096
```

23 Advanced Bacula Backup Proxy Pod Deployment

Warning: This is an advanced topic related to OpenShift clusters. You should **NOT** try to implement or customize the Bacula OpenShift Plugin behavior unless you REALLY know what you are doing.*

You can customize the service parameters used for deploying Bacula backup Pods dedicated to Persistent Volume Claim data backup to suit your needs. The plugin uses the following Pod service deployment YAML template to execute the proxy operation pod on the cluster.

```
apiVersion: v1
kind: Pod
metadata:
  name: {podname}
  namespace: {namespace}
  labels:
    app: {podname}
spec:
  hostname: {podname}
```

(continues on next page)

(continued from previous page)

```
  {nodenameparam}
  containers:
  - name: {podname}
    resources:
      limits:
        cpu: "1"
        memory: "64Mi"
      requests:
        cpu: "100m"
        memory: "16Mi"
    image: {image}
    env:
    - name: PLUGINMODE
      value: "{mode}"
    - name: PLUGINHOST
      value: "{host}"
    - name: PLUGINPORT
      value: "{port}"
    - name: PLUGINTOKEN
      value: "{token}"
    imagePullPolicy: {imagepullpolicy}
    volumeMounts:
      - name: {podname}-storage
        mountPath: /{mode}
  restartPolicy: Never
  volumes:
    - name: {podname}-storage
      persistentVolumeClaim:
        claimName: {pvcname}
```

The above template uses a number of predefined placeholders which will be replaced by corresponding variables during Pod execution preparation. To customize proxy Pod deployment you can change or tune template variables or the template body. Below is a list of all supported variables with short descriptions and requirement conditions.

podname This is the predefined Pod name used by a plugin. This variable is required and cannot be customized.

namespace This is a Namespace name for which the PVC Data backup is performed. This variable is required and cannot be customized.

nodenameparam This is a placeholder for Cluster node name parameter (`nodeName: ...`) used to mount an existing Volume Claim for backup or restore if required for the selected PVC. In most cases this variable is required and cannot be customized.

image This is a Pod Container image name to execute. You can customize or omit this variable as long as you provide a Container image name required by the cluster.

mode This is an operation mode for the Proxy Pod. The supported values are: `backup` and `restore`. This variable is required and cannot be customized.

host This is an endpoint address which corresponds to the `pluginport=...` OpenShift plugin parameter. This variable is required. You can customize or omit this variable as long as you provide a value for the **PLUGINHOST** Container environment.

port This is an endpoint address which corresponds to the `pluginport=...` Kubernetes plugin parameter. This variable is required. You can customize or omit this variable as long as you provide a value for the **PLUGINPORT** Container environment.

token This is an Authorization Token (randomly generated). This variable is required and cannot be customized.

pvcname This is the name of the PVC for backup or restore operations. This variable is required and cannot be customized.

You can create the required file: `/opt/bacula/scripts/bacula-backup.yaml` or point to the custom one using the `$DEFAULTPODYAML` environment variable.

24 Limitations

- Only full level backups are possible. This is an OpenShift limitation.
- You can perform a single PVC Data backup or restore with a single Bacula File Daemon installation associated with single **fdaddress=<name>**. This limitation may be removed in a future release of the OpenShift plugin.
- The `restart` command has limitations with plugins, as it initiates the Job from scratch rather than continuing it. Bacula determines whether a Job is restarted or continued, but using the `restart` command will result in a new Job.

25 Common Problems

In this section we describe the common problems you may encounter when you first deploy the Bacula OpenShift plugin or when you are not very familiar with this plugin.

- You have set the **incluster** parameter in your Job but you have the following error:

```
Error: openshift: incluster error: Service host/port is not set.
```

  This means you are running the Bacula File Daemon and OpenShift plugin not in a Pod, or Kubernetes does not provide default service access in your installation. In the latter case you should use a standard OpenShift access method in a prepared kubeconfig file.

