Service Backups SDK 18-4

Service Backups SDK 18-4

You can find the most up-to-date technical documentation on the VMware website at:
https://docs.vmware.com/

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Service Backups for Pivotal Cloud Foundry

BOSH operators running services (e.g. Redis service broker for Cloud Foundry) may want to back up certain files from the virtual machines running these services so that they can restore them after a disaster.

Configuration

The Service Backup BOSH release backs up a directory on the instance VM it is located on to one of several supported destination types. The supported destination types are AWS S3, Azure blobstore, and SCP.

Uploading a Service Backup Release

Service Backup is distributed as a BOSH final release. To upload a release to your BOSH director, upload the latest final release tarball from VMware Tanzu Network.

Configuring the Manifest

Service Backup is designed to be colocated on service instance VMs, and must be included in that service's BOSH deployment manifest.

Add Service Backups to Your Service Deployment

Shown below is an template manifest, adding an S3 backup destination to a Redis service deployment. For further information on changing the backup destination, see link.

```
---
properties:
  service-backup:
    destinations:
    - type: s3
      name: <OPTIONAL: destination name>
      config:
        bucket_name: <bucket>
        bucket_path: <path in bucket>
        access_key_id: <aws access key>
        secret_access_key: <aws secret key>
        endpoint_url: <OPTIONAL: S3 compatible endpoint URL>
        region: <OPTIONAL: S3 region, required for Signature Version 4 regions>
    source_folder: <directory to back up>
    cron_schedule: <cron schedule>
    backup_user: <OPTIONAL: backup user>
    source_executable: <OPTIONAL: run before each backup>
    exit_if_in_progress: <OPTIONAL: exits if another backup is already running. defaul
ts to false>
```

```
    missing_properties_message: <OPTIONAL: message to log when properties are missing
in the manifest. defaults to 'Provide these missing fields in your manifest.'>
    service_identifier_executable: <OPTIONAL: command that prints service instance ID
on stdout>
    cleanup_executable: <OPTIONAL: command to run after each backup>
    add_deployment_name_to_backup_path: <OPTIONAL: if set true will include deployment
 name in destination path. Defaults to false>
    alerts: # optional
      product_name: <product name>
      config:
        cloud_controller:
          url: <Cloud Foundry API URL>
          user: <Cloud Foundry username with SpaceAuditor role in cf_space>
          password: <Cloud Foundry password>
        notifications:
          service_url: <Cloud Foundry notification service URL>
          cf_org: <Cloud Foundry org name>
          cf_space: <Cloud Foundry space name>
          reply_to: <OPTIONAL: email reply-to address. This is required for some SMTP
servers>
          client_id: <UAA client ID with authorities to send notifications>
          client_secret: <UAA client secret>
        timeout_seconds: <OPTIONAL: default is 60>
        skip_ssl_validation: <OPTIONAL: ignore TLS certification verification errors>

releases:
- name: redis
  version: latest
- name: service-backup
  version: latest

instance_groups:
- name: redis-server
  jobs:
  - name: redis-server
    release: redis
  - name: service-backup
    release: service-backup
```

Configure the Backup Schedule

Backups will be triggered according to the schedule given by the `cron_schedule` property. See robfig/cron for cron expression syntax.

Configure the Backup User

By default the service backup process will run as ‘vcap’. This can be configured by setting the `backup_user` property.

If you are also providing your own `source_executable` be sure that the `backup_user` you create has execute permissions for it.

Defining the Files to Be Backed Up

The `source_folder` property names a local path from which backups are uploaded. All files in this location are uploaded, therefore, Pivotal recommends that this directory be separate from the

directory that files are written to. This avoids uploading files that are still being written.

Preparing the Files to be Backed Up

The `source_executable` is an optional property that names a command to run before each backup. Any required arguments can be provided space-separated after the executable name. The property is useful for services that require some operation to be performed before backing files up, for example triggering a Redis memory dump to disk. The service author may also use the `source_executable` to decide which instance performs the backup. For example, the service author can choose to only trigger backups on `replica` nodes to avoid issues with data consistency.

If the property is not specified, it is ignored and nothing is executed.

If the property is specified, when the BOSH lifecycle runs stop scripts, any running processes of `source_executable` will be identified and will be sent the SIGTERM signal. This attempts to stop the main service-backup process, which first forwards the signal to any active backup processes and allows them to trap the signal and clean up any used resources, such as open files, before potential removal or update of VMs. If the main service-backup process does not terminate successfully with the SIGTERM signal within 15 seconds, SIGQUIT then SIGKILL signals are sent respectively and the main service-backup process is terminated by force. In the event that SIGQUIT or SIGKILL is sent, the signal will not be forwarded to active backup processes, but no new backup processes will be initiated by the main process.

If a suitable executable is not included in the service release, you can add one by publishing it in a separate release, as its own package and job, and colocating it into the deployment.

Cleaning Up the Files Which Were Backed Up

The optional `cleanup_executable` property names a local executable to cleanup backups. Tokens are split on spaces; first is command to execute and remaining are passed as args to command.

Sending an Alert When a Backup Fails

When the optional `alerts` properties are configured an alert will be sent to SpaceDevelopers in the configured `cf_space` when a backup fails. Alerts are sent using the Cloud Foundry Notifications Service.

Correlating BOSH Instances to Cloud Foundry Service Instances

BOSH operators might want to correlate BOSH-deployed VM instances with CF service instances, in which case the service author must provide a binary that returns a string identifier for your service instance. This will appear in all log messages under the data element `identifier`. For example:

```
{ "source": "ServiceBackup", "message": "doing-stuff", "data": { "backup_guid":"244eadb0-91e7-45da-9a7f-3616a59a6e61", "identifier": "service_identifier" }, "timestamp": "2021-04-05T16:28:16.000000000Z", "log_level": "info" }
```

Add the optional `service_identifier_executable` key to your manifest (tokens are split on spaces; first is command to execute and remaining are passed as args to command).

Naming Backup Destinations

Each destination can be given an optional `name` property. This will appear in the log messages for uploads to that destination. For example:

```
{ "timestamp":"2021-04-05T16:28:16.000000000Z", "source":"ServiceBackup", "message":"S
erviceBackup.WithIdentifier.about to upload /path/to/file to S3 remote path bucket_nam
e/2016/07/04", "log_level":"info", "data": { "backup_guid":"244eadb0-91e7-45da-9a7f-36
16a59a6e61", "destination_name": "some-destination-name", "identifier": "service_ident
ifier", "session":"1" } }
```

Identifying Logs for a Backup Run

The log lines of a particular backup run can be identified by correlating their unique `backup_guid` For example

```
{"timestamp":"2021-04-05T16:28:16.000000000Z","source":"ServiceBackup","message":"Serv
iceBackup.WithIdentifier.Upload backup started","log_level":"info","data":{"backup_gui
d":"244eadb0-91e7-45da-9a7f-3616a59a6e61","identifier":"service_identifier","session":
"1"}}
{"timestamp":"2021-04-05T16:28:17.000000000Z","source":"ServiceBackup","message":"Serv
iceBackup.WithIdentifier.Upload backup completed successfully","log_level":"info","dat
a":{"backup_guid":"244eadb0-91e7-45da-9a7f-3616a59a6e61","duration_in_seconds":8.08100
0000000001e-06,"identifier":"service_identifier","session":"1","size_in_bytes":200}}
{"timestamp":"2021-04-05T16:28:17.000000000Z","source":"ServiceBackup","message":"Serv
iceBackup.WithIdentifier.Cleanup started","log_level":"info","data":{"backup_guid":"24
4eadb0-91e7-45da-9a7f-3616a59a6e61","identifier":"service_identifier","session":"1"}}
{"timestamp":"2021-04-05T16:28:17.000000000Z","source":"ServiceBackup","message":"Serv
iceBackup.WithIdentifier.Cleanup debug info","log_level":0,"data":{"backup_guid":"244e
adb0-91e7-45da-9a7f-3616a59a6e61","cmd":"creator-cmd","identifier":"service_identifier
","out":"Cleanup Complete\n","session":"1"}}
```

Triggering Manual Service Backups

BOSH operators might want to trigger a one-off, manual backup. To do this:

1. ssh onto a BOSH-deployed VM that has a service-backup job running on it.
2. Execute `/var/vcap/jobs/service-backup/bin/manual-backup`.

Disabling Service Backups

Backups can be disabled by removing the `service-backup` section from your manifest and then redeploying. You can still leave the job on your instance group if you wish.

Backup Destinations

Service Backup supports S3 (AWS, Ceph s3, Swift w/ S3 compatibility module), Azure blobstore, and SCP. To change the backup destination change the manifest `destinations` value:

S3

```
properties:
  service-backup:
```

```
    destinations:
    - type: s3
      name: <OPTIONAL: destination name>
      config:
        bucket_name: <bucket>
        bucket_path: <path in bucket>
        access_key_id: <aws access key>
        secret_access_key: <aws secret key>
        endpoint_url: <OPTIONAL: url for S3-compatible blobstore>
        region: <OPTIONAL: S3 region, required for Signature Version 4 regions>
```

New S3 Buckets

If the bucket does not exist in S3, then it will be created in the `us-east-1` region. To create the bucket in another region, configure the `region` property.

Existing S3 Buckets

If the bucket exists in S3 and requires the Signature Version 4 Signing Process then the `region` property must be configured. Some S3 regions require Signature Version 4, e.g. `eu-central-1`. To obtain a bucket's region run `aws s3api get-bucket-location --bucket BUCKET_NAME`.

AWS IAM

If you are using AWS ensure that the IAM user has the right permissions. Create a new custom policy (IAM > Policies > Create Policy > Create Your Own Policy) and paste in the following permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ServiceBackupPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts",
        "s3:CreateBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::MY_BUCKET_NAME/*",
        "arn:aws:s3:::MY_BUCKET_NAME"
      ]
    }
  ]
}
```

The `s3:CreateBucket` permission is required because the tool will attempt to create the bucket if it does not already exist. If the desired bucket already exists, the `s3:CreateBucket` permission is not required.

Finally, attach this policy to your AWS user (IAM > Policies > Policy Actions > Attach).

AWS CLI version

The current release uses the `aws` CLI version `aws-cli/1.11.91 Python/3.6.1 Darwin/15.6.0 botocore/1.5.54` to upload.

S3-compatible blobstores

By default, backups are sent to AWS S3. To use an S3-compatible blobstore like RiakS2, set the `endpoint_url` property.

Service Backup uses the AWS CLI to backup files to S3, or S3-compatible blobstores. If the endpoint has a self-signed SSL server certificate, then the root CA certificate must be added to the default system trust store. This can be done using BOSH trusted certs.

Azure

```
properties:
  service-backup:
    destinations:
    - type: azure
      name: <OPTIONAL: destination name>
      config:
        storage_account: <storage account>
        storage_access_key: <storage key>
        container: <container name>
        path: <path in container>
        endpoint: <OPTIONAL: Azure Storage endpoint>
```

By default, backups are sent to the public Azure blobstore. If you are not using an Azure public region, you can specify a different endpoint name using the `endpoint` property.

Azure client version

The current release uses `blobxfer` CLI version `1.0.0` to upload.

Google Cloud Storage

```
properties:
  service-backup:
    destinations:
    - type: gcs
      name: <OPTIONAL: destination name>
      config:
        service_account_json: |
          <GCP service account key JSON literal>
        project_id: <GCP project ID>
        bucket_name: <GCP Storage bucket name, does not have to already exist>
```

The service account must have “Storage Admin” IAM permissions. You can generate the JSON key for a service account with:

```
gcloud iam service-accounts keys \
```

```
  create key-lives-here.json \
  --iam-account $IAM_ACCOUNT_ADDRESS
```

If the bucket does not already exist, service-backup will create it. It uses the API default attributes for the bucket, summarised here: Bucket ACL rules: for the project that owns the configured service account, owners own the bucket, editors can write the bucket (CRUD objects), viewers can read the bucket. Objects in bucket ACL rules: for the project that owns the configured service account, owners and editors can read/write/delete the object, viewers can read the object. Location: US Storage class: Standard. See storage classes documentation.

If you create the bucket in advance, then you must ensure that the service account has access to write it. "Storage Admin" IAM permission should ensure this.

Google Cloud Storage version

The current release uses the Google GCS storage golang package at `commit 86c12b7` to upload.

SCP

```
properties:
  service-backup:
    destinations:
    - type: scp
      name: <OPTIONAL: destination name>
      config:
        user: <ssh username>
        server: <ssh server>
        destination: <path to upload to on server>
        fingerprint: <host-fingerprint> #optional
        key: |
          -----BEGIN EXAMPLE RSA PRIVATE KEY-----
            ...
          -----END EXAMPLE RSA PRIVATE KEY-----
        port: <optional ssh port. Defaults to 22>
```

The `fingerprint` field expects the entire output in the format returned by the `ssh-keyscan` utility for the host. If the fingerprint is provided and doesn't match, then the backup will fail. If it's empty then the fingerprint of the host will be requested right before the upload and this would be used instead. A fingerprint should be configured to prevent server spoofing or man-in-the-middle attacks. For more information refer: http://man.openbsd.org/ssh#authentication

SCP version

The current release leverages the `scp` that is included in the stemcell. Check your stemcell for its `scp` version.

Multiple destinations

```
properties:
  service-backup:
    destinations:
    - type: s3
```

```
    name: <OPTIONAL: destination name>
    config:
      bucket_name: <bucket>
      bucket_path: <path in bucket>
      access_key_id: <aws access key>
      secret_access_key: <aws secret key>
  - type: scp
    name: <OPTIONAL: destination name>
    config:
      user: <ssh username>
      server: <ssh server>
      destination: <path to upload to on server>
      key: |
        -----BEGIN EXAMPLE RSA PRIVATE KEY-----
          ...
        -----END EXAMPLE RSA PRIVATE KEY-----
      port: <optional ssh port. Defaults to 22>
```

The tool can be provided with configuration for multiple destinations in the `destinations` property. The tool will upload backups to all the provided destinations sequentially.

You can configure multiple destinations of the same type, for example: two S3 buckets in different regions.

Operating

Locating the Backups

If `add_deployment_name_to_backup_path` is configured `true` the tool will add the deployment name in your destination bucket / folder, for example `<bucket-name>/<bucket-path>/<deployment-name>/*`.

The tool will then create a date-based folder structure in your destination bucket / folder as follows: `<bucket-name>/<bucket-path>/YYYY/MM/DD/` or `<bucket-name>/<bucket-path>/<deployment-name>/YYYY/MM/DD/`. The tool uses the BOSH VM it is running on to calculate the date, so for example if your VM is using UTC time, then the folder structure will reflect this.

For example, on S3 the provided path is appended with the current date such that the resultant path is `/my/remote/path/inside/bucket/YYYY/MM/DD/` and hence the backups are accessible at `s3://my-bucket-name/my/remote/path/inside/bucket/YYYY/MM/DD/`.

If `add_deployment_name_to_backup_path` is configured `true` and the deployment is called `deployed-service`, the resultant path will be `/my/remote/path/inside/bucket/deployed-service/YYYY/MM/DD/` and the backups are accessible at `s3://my-bucket-name/my/remote/path/inside/bucket/deployed-service/YYYY/MM/DD/`.

Logging

Service backup logs to files in `/var/vcap/sys/log/service-backup`, and also to syslog.

For forwarding syslog to a third party syslog drain (e.g. papertrail), Pivotal recommends colocating the syslog-release..

Monitoring

Here are log messages you may choose to monitor for. The log messages appear on one line and

have been formatted here for easy reading.

Internal Checksums

If you are publishing a tile to be consumed by Ops Manager 1.8.x or 1.9.x, you will need to build your tile using releases with SHA1 internal checksums. Service Backup releases are published using SHA2 internal checksums. You can convert these releases to use SHA1 internal checksums using the BOSH CLI command `sha1ify-release`.

Troubleshooting

ServiceBackup.Error scheduling job

This error occurs when the cron schedule is invalid, e.g. "* * * * * 99":

```
{
  "timestamp": "2021-04-05T16:28:16.000000000Z",
  "source": "ServiceBackup",
  "message": "ServiceBackup.Error scheduling job",
  "log_level": "error",
  "data": {
    "error": "End of range (99) above maximum (6): 99"
  }
}
```

ServiceBackup.No destination provided - skipping backup

This warning will be logged when no destination is provided.

```
{
  "timestamp": "2021-04-05T16:28:16.000000000Z",
  "source": "ServiceBackup",
  "message": "ServiceBackup.No destination provided - skipping backup",
  "log_level": "info",
  "data": {}
}
```

ServiceBackup.Perform backup completed with error

This error will be logged when performing a backup fails, e.g. when the backup command exits status 1:

```
{
  "timestamp": "2021-04-05T16:28:16.000000000Z",
  "source": "ServiceBackup",
  "message": "ServiceBackup.Perform backup completed with error",
  "log_level": "error",
  "data": {
    "backup_guid": "3ebc4d08-62a3-4919-a02c-841c1afe51d0",
    "error": "exit status 1"
  }
}
```

ServiceBackup.Upload backup completed with error

This error will occur when uploading a backup fails. For example, when the S3 credentials provided are not authorised to create a bucket:


```
{
  "timestamp": "2021-04-05T16:28:16.000000000Z",
  "source": "ServiceBackup",
  "message": "ServiceBackup.Upload backup completed with error",
  "log_level": "error",
  "data": {
    "backup_guid": "3ae36910-3c1d-4e53-b28e-62105aee1e79",
    "error": "error in create bucket: exit status 1, output: make_bucket failed: s3://
doesnotexist5f7c0f16-bba4-49c9-9f60-371366edea3b An error occurred (AccessDenied) when
 calling the CreateBucket operation: Access Denied\n"
  }
}
```


And when the SCP host fingerprint is invalid:


```
{
  "timestamp": "2021-04-05T16:28:16.000000000Z",
  "source": "ServiceBackup",
  "message": "ServiceBackup.Upload backup completed with error",
  "log_level": "error",
  "data": {
    "backup_guid": "7bd784ae-2375-43fd-bd9b-b4a43c2368c2",
    "error": "error checking if remote path exists: 'exit status 255', output: 'No ECD
SA host key is known for localhost and you have requested strict checking.\r\nHost key
 verification failed.\r\n'"
  }
}
```


ServiceBackup.Backup currently in progress

This error will occur when the tool is configured with `exit_if_in_progress: true` and a backup starts whilst another backup is in progress.


```
{
  "timestamp": "2021-04-05T16:28:16.000000000Z",
  "source": "ServiceBackup",
  "message": "ServiceBackup.Backup currently in progress, exiting. Another backup will
 not be able to start until this is completed.",
  "log_level": "error",
  "data": {
    "backup_guid": "94ca42ed-c289-4b15-a76a-23fe55d4955b",
    "error": "backup operation rejected"
  }
}
```


ServiceBackup.Error running

An unexpected error has occurred.

Create a pull request or raise an issue on the source for this page in GitHub

Service Backups for PCF Release Notes

For product versions and upgrade paths, see Upgrade Planner.

v18.4.0

Release Date: May 13, 2021

New Features

New features and changes in this release:

- **Timestamps:** Now logged in a human-readable format.
- **Go:** Updated to v1.16.3.

Resolved Issues

This release has the following fix:

- **Fixes an issue with alerts:** Service Backup no longer attempts to send alerts if you have not provided alerting configuration.

Known Issues

There are no known issues for this release.

View Release Notes for Another Version

To view the release notes for another product version, select the version from the dropdown at the top of this page.

Create a pull request or raise an issue on the source for this page in GitHub