Introduction

This document contains best practices to consider when defining your backup and restore procedure.

For Docker Enterprise Edition, backups should be done separately for each components of the platform:

- Docker Swarm
- UCP
- DTR

> To recover Docker Enterprise Edition in high availability mode, since the components work in cluster mode (UCP cluster and DTR cluster), remove unhealthy nodes and join new ones to bring it back in a healthy state. For this cluster mode, only use the backup as a last resort.

General Prerequisites

Before performing the backup or restore process for Docker EE, you must meet the following requirements:

- Start with healthy managers, otherwise it's a disaster recovery (consider reaching out to Docker Support for next steps).
- UCP/DTR/Engine versions must be the same for backup and for restore.
- Swarm managers **must** be restored on nodes with the same IP address.

  > Avoid swarm backup by storing stacks, services definitions, secrets, and networks definitions in a *Source Code Management* or *Config Management* tool.

- Running Stacks/Services are not impacted by the backups.
- Your application data is **not** backed up. It must be managed separately.

Backup Procedure

Your backup policy should include regularly-scheduled backups of Swarm, UCP, and DTR independently.

Things to consider when performing a backup:

- Since Swarm and UCP store the same data on all manager nodes, you only need to take periodic backups of a single manager node.
- For DTR, you should always create backups from the same DTR replica to ensure a smoother restore.
- UCP UI will have warnings during Swarm and UCP backup.
- Since the UCP manager stops during the backup, there is a short period of time where the cluster runs with one less manager. Consider running a 5-manager cluster so that when you lose one while the backup is running, you won't experience a disruption in services.

Swarm Backup

Backup your Swarm using any manager. This manager will not be available during the backup process, so expect downtime of the manager and plan accordingly.

1. Retrieve your Swarm unlock key if `auto-lock` is enabled.
2. Stop the Docker daemon to prevent any changes during the backup.
3. Back up the entire `/var/lib/docker/swarm` directory.
4. Restart the manager.

> See the Docker docs (https://docs.docker.com/engine/swarm/swarm_manager_locking/) for more information about the Swarm lock feature.

Data	Description	Backed up
Raft keys	Used to encrypt communication among Swarm nodes and to encrypt and decrypt Raft logs	yes
membership	List of the nodes in the cluster	yes
Services	Stacks and services stored in Swarm-mode	yes
(overlay) networks	The overlay networks created on the cluster	yes
configs	The configs created in the cluster	yes
secrets	Secrets saved in the cluster	yes
Swarm unlock key	**Must be saved on a password manager !**	no

See the full documentation for Swarm backups on docs.docker.com (https://docs.docker.com/engine/swarm/admin_guide/#back-up-the-swarm).

UCP Backup

Perform a UCP backup on a manager node. This manager will not be available during the backup process, so plan accordingly. High availability is mandatory to avoid downtime.

1. Create a backup of a UCP manager node (replace `<UCP_VERSION>` with the version you are currently running):

```
$ docker container run --log-driver none --rm -i --name ucp \
  -v /var/run/docker.sock:/var/run/docker.sock \
  docker/ucp:<UCP_VERSION> backup --interactive > /tmp/backup.tar
```

2. In a valid backup file, over 100 files should appear in the list and the `./ucp-node-certs/key.pem` file should be present. Ensure the backup is a valid tar file by listing its contents.

```
$ tar --list -f /tmp/backup.tar
```

3. Optionally, you can encrypt the backup using a passphrase (replace `<UCP_VERSION>` with the version you are currently running):

```
$ docker container run --log-driver none --rm -i --name ucp \
  -v /var/run/docker.sock:/var/run/docker.sock \
  docker/ucp:<UCP_VERSION> backup --interactive \
  --passphrase "secret" > /tmp/backup.tar

# Decrypt the backup and list its contents
$ gpg --decrypt /tmp/backup.tar | tar --list
```

For Docker EE 17.06 or higher, if the Docker engine has SELinux enabled, which is typical for RHEL hosts, you need to include `--security-opt label=disable` in the `docker` command (replace `<UCP_VERSION>` with the version you are currently running):

```
$ docker container run --security-opt label=disable --log-driver none --rm -i --name ucp \
  -v /var/run/docker.sock:/var/run/docker.sock \
  docker/ucp:<UCP_VERSION> backup --interactive > /tmp/backup.tar
```

> To find out whether SELinux is enabled in the engine, view the host's `/etc/docker/daemon.json` file, and search for the string `"selinux-enabled":"true"`.

Data	Description	Backed up
Configurations	The UCP configurations, including Docker EE license. Swarm and client CAs	yes
Access control	Permissions for teams to swarm resources, including collections, grants, and roles	yes
Certificates and keys	Certificates and keys used for authentication and mutual TLS communication	yes
Metrics data	Monitoring data gathered by UCP	yes
Organizations	Your users, teams, and orgs	yes
Volumes	All UCP named volumes, which include all UCP component certs and data	yes
Overlay Networks	Swarm-mode overlay network definitions	no
Configs, Secrets	Create a Swarm backup to backup these data	no
Services	Stacks and services are stored in Swarm-mode or SCM/Config Management	no

See the UCP backup documentation on docs.docker.com (https://docs.docker.com/datacenter/ucp/2.2/guides/admin/backups-and-disaster-recovery/#data-managed-by-ucp).

DTR Backup

The backup command doesn't cause downtime for DTR, so you can take frequent backups without affecting your users.

The following example shows how to create a backup of DTR metadata on a manager or a replica node (replace `<DTR_VERSION>` with the version you are currently running):

```
$ docker run --log-driver none -i --rm \
  docker/dtr:<DTR_VERSION> backup \
  --ucp-url <UCP_URL> \
  --ucp-insecure-tls \
  --ucp-username <UCP_USERNAME> \
  --ucp-password <UCP_PASSWORD> \
  --existing-replica-id <REPLICA_ID> > backup-metadata.tar
```

The backup contains private keys and sensitive data, so you can encrypt the resulting tar file with a passphrase by running:

```
$ gpg --symmetric backup-metadata.tar
```

You can test the backup and validate it with the following:

```
$ tar -tf backup-metadata.tar

# The archive should look like this
dtr-backup-v2.4.1/
dtr-backup-v2.4.1/rethink/
dtr-backup-v2.4.1/rethink/properties/
dtr-backup-v2.4.1/rethink/properties/0
```

With an encrypted tarball, use the following:

```
$ gpg -d backup-metadata.tar | tar -t
```

If you've configured DTR to store images on the local filesystem or NFS mount, backup the images by using ssh to log into a node where DTR is running and creating a tar archive of the `dtr-registry` volume.

> It is not recommended to store images on the local filesystem on production, NFS is recommended instead.

```
$ sudo tar -cf backup-images.tar \
  $(dirname $(docker volume inspect --format '{{.Mountpoint}}' dtr-registry-<REPLICA-ID>))
```

If you're using a different storage backend, follow the best practices recommended for that system.

Data	Description	Backed up
Configurations	DTR settings	yes
Repository metadata	Metadata like image architecture and size	yes
Access control to repos and images	Data about who has access to which images	yes
Notary data	Signatures and digests for images that are signed	yes
Scan results	Information about vulnerabilities in your images	yes
Certificates and keys	TLS certificates and keys used by DTR	yes
Image content	Needs to be backed up separately, depends on DTR configuration	no

Users, orgs, teams	Create a UCP backup to backup this data	no
Vulnerability database	Can be re-downloaded after a restore	no

See the Docker docs (https://docs.docker.com/datacenter/dtr/2.4/guides/admin/backups-and-disaster-recovery/#data-managed-by-dtr) for more information about data managed by DTR.

Restore Procedure

This section assumes you are using a new cluster. Otherwise, you need to uninstall all the previous installation first. See DTR Remove Documentation (https://docs.docker.com/datacenter/dtr/2.4/reference/cli/remove/).

Remember, to restore Swarm from a backup, you need a node with the **same** IP address as the backed up one.

You can find the list of managers IP addresses in the file `state.json` in the zip file.

Swarm Restore

If `auto-lock` was enabled on the old Swarm, the unlock key will be required to perform the restore.

1. Shut down Docker on the target host machine where the swarm will be restored.
2. Remove the contents of the `/var/lib/docker/swarm` directory on the new Swarm if it exists.
3. Restore the `/var/lib/docker/swarm` directory with the contents of the backup.
4. Start Docker on the new node.
5. Unlock the Swarm if necessary.
6. Re-initialize the swarm using the following command, so that this node does not attempt to connect to nodes that were part of the old Swarm, and presumably no longer exist:

   ```
   $ docker swarm init --force-new-cluster
   ```

7. Verify that the state of the Swarm is as expected. This may include application-specific tests or simply checking the output of `docker service ls` to be sure that all expected services are present.
8. If you use `auto-lock`, rotate the unlock key.

UCP Restore

There are two ways to restore UCP:

1. On a manager node of an existing Swarm which does not have UCP installed. In this case, UCP restore will use the existing Swarm.
2. On a Docker engine that isn't participating in a Swarm. In this case, a new Swarm is created, and UCP is restored on top.

You only need to be sure that UCP is not running on the manager you are trying to restore. If UCP is running, you should uninstall (https://docs.docker.com/datacenter/ucp/2.2/guides/admin/install/uninstall/) it first.

The following example shows how to restore UCP from an existing backup file, presumed to be located at `/tmp/backup.tar` (replace `<UCP_VERSION>` with the version of your backup):

```
$ docker container run --rm -i --name ucp \
  -v /var/run/docker.sock:/var/run/docker.sock  \
  docker/ucp:<UCP_VERSION> restore < /tmp/backup.tar
```

If the backup file is encrypted with a passphrase, provide the passphrase to the restore operation(replace `<UCP_VERSION>` with the version of your backup):

```
$ docker container run --rm -i --name ucp \
  -v /var/run/docker.sock:/var/run/docker.sock  \
  docker/ucp:<UCP_VERSION> restore --passphrase "secret" < /tmp/backup.tar
```

The restore command may also be invoked in interactive mode, in which case the backup file should be mounted to the container rather than streamed through stdin (replace `<UCP_VERSION>` with the version of your backup):

```
$ docker container run --rm -i --name ucp \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v /tmp/backup.tar:/config/backup.tar \
  docker/ucp:<UCP_VERSION> restore -i
```

After you successfully restored UCP, you can add new managers and workers the same way you would after a fresh installation.

DTR Restore

You only need to restore a replica from a backup if your DTR has a majority of unhealthy replicas. Otherwise, you can `remove` the unhealthy replica and use the `join` command to add it again to the cluster.

For a DTR restore:

- UCP must be up and running and healthy. You do not have to stop any running replicas.
- You must restore DTR on the same UCP cluster where you've created the backup. **Do not try** to restore on a different UCP cluster; all DTR resources will be owned by users that don't exist, so you'll not be able to manage the resources, even though they're stored in the DTR data store.

You can restore the DTR metadata with the `docker/dtr restore` command. This performs a fresh installation of DTR and reconfigures it with the configuration created during a backup (replace `<DTR_VERSION>` with the version of your backup):

```
$ docker run -i --rm \
  docker/dtr:<DTR_VERSION> restore \
  --ucp-url <UCP_URL> \
  --ucp-insecure-tls \
  --ucp-username <UCP_USERNAME> \
  --ucp-node <HOSTNAME> \
  --replica-id <REPLICA_ID> \
  --dtr-external-url <DTR_EXTERNAL_URL> < backup-metadata.tar
```

If you are using a NFS backend storage for the images data, add the argument `--nfs-storage-url <NFS_STORAGE_URL>` on the restore command line. This will restore the metadata with your NFS backend configured.

If you're scanning images, you now need to download the vulnerability database. The vulnerability database is not part of the DTR backup.

If you had DTR configured to store images on the local filesystem (not recommended for production), you can extract your backup:

```
$ tar -xzf backup-images.tar -C /var/lib/docker/volumes
```

If you're using a different storage backend, follow the best practices recommended for that system. When restoring the DTR metadata, DTR will be deployed with the same configurations it had when creating the backup.

> If you are using a NFS backend storage for the Images Data, add the argument `--nfs-storage-url <NFS_STORAGE_URL>` on the restore command line. This will restore the metadata with your NFS backend configured.

After you successfully restore DTR, you can join new replicas the same way you would after a fresh installation.