Abstract

A guide for the "atomic" command-line tool
# Table of Contents

CHAPTER 1. OVERVIEW .................................................................................................. 3.

CHAPTER 2. PREREQUISITES ...................................................................................... 4.

CHAPTER 3. ATOMIC COMMANDS .............................................................................. 5.

3.1. ATOMIC HOST .......................................................... 5
3.2. ATOMIC DIFF .......................................................... 6
3.3. ATOMIC INSTALL ...................................................... 7
3.4. ATOMIC UNINSTALL ................................................ 8
3.5. ATOMIC RUN ............................................................ 8
3.6. ATOMIC STOP .......................................................... 8
3.7. ATOMIC IMAGES ...................................................... 8
3.8. ATOMIC CONTAINERS ............................................. 11
3.9. ATOMIC TOP ............................................................ 12
3.10. ATOMIC MOUNT .................................................... 13
3.11. ATOMIC UNMOUNT ............................................. 13
3.12. ATOMIC PULL ........................................................ 13
3.13. ATOMIC PUSH ........................................................ 14
3.14. ATOMIC STORAGE (MIGRATE) ................................. 14
3.15. ATOMIC SCAN ....................................................... 15
3.16. ATOMIC SIGN ....................................................... 17
3.17. ATOMIC TRUST ...................................................... 18
3.18. ATOMIC UPDATE .................................................. 18
3.19. ATOMIC --HELP AND MANUAL PAGES ....................... 19
CHAPTER 1. OVERVIEW

The atomic command-line tool provides a way to interact and manage Atomic Host systems and containers. It provides a high level, coherent entrypoint to the system and makes it easier to interact with special kinds of containers, such as super-privileged containers, and debugging tools.

The atomic command uses tools such as docker, ostree and skopeo to manage containers and container host systems. There are also a lot of features built into the atomic command that are not available in the docker command. These features allow you to use special commands for image signing, image verification, the ability to install a container - mounting file systems and opening privileges.

Understanding LABELs: Dockerfiles support storing default values for some commands that atomic can read and execute. These are called "LABEL" instructions and they make it easy to ship images with their own suggested values, and simplifies running complex docker commands. For example, if a Dockerfile contains the LABEL RUN, running atomic run <image> executes its contents. The commands in atomic that use labels are install, uninstall, mount, unmount, run, and stop.
CHAPTER 2. PREREQUISITES

- On **RHEL Atomic Host**, `atomic` is part of the OSTree and is ready to use.
- On **Red Hat Enterprise Linux**, make sure you have covered the following:
  - Subscribe the system to the Extras channel which provides the `atomic` package.

For Red Hat Subscription Management run this command:

```bash
# subscription-manager repos --enable rhel-7-server-extras-rpms
```

If you are using Satellite, run:

```bash
# rhn-channel --add --channel rhel-x86_64-server-extras-7
```

- Install `atomic` using Yum:

```bash
# yum install atomic
```

- Make sure the **docker service** is running:

```bash
# systemctl status docker
```

If the output states "inactive", use the following command:

```bash
# systemctl start docker
```

**Note**

On both systems, you need to have root privileges to use `atomic`. 
CHAPTER 3. ATOMIC COMMANDS

3.1. ATOMIC HOST

This subcommand is a high-level wrapper for the `rpm-ostree`, tool which performs upgrades, rollbacks, and system state inspection.

- `atomic host status`

Lists information about all deployments. The asterisk (*) marks the currently running deployment.

```
# atomic host status
State: idle
Deployments:
* rhel-atomic-host-ostree:rhel-atomic-host/7/x86_64/standard
  Version: 7.3 (2016-09-27 17:53:07)
  BaseCommit:
  Commit:
    d3fa3283db85ee656f78dcfc0fcffe6cd5aa06596dac6ec5e436352298a59cb
  OSName: rhel-atomic-host

  rhel-atomic-host-ostree:rhel-atomic-host/7.2/x86_64/autobrew/buildmaster
  Version: 7.2.7 (2016-09-15 22:28:54)
  BaseCommit:
  Commit:
    dbbc8e805f003d8e56658dc220f1fe1397caf80221cc050eeb1baf44be56a1
  OSName: rhel-atomic-host
```

- `atomic host rollback`

Switches to the other installed tree at the next boot. You can use the `-r` option to initiate a reboot after rollback is prepared:

```
# atomic host rollback -r
```

- `atomic host upgrade`

Upgrades to the latest OSTree if available. This can take a few minutes. When done, it gives you a full list of `changed`, `removed`, and `added` packages. The newly downloaded tree boots automatically at next reboot.

- `atomic host deploy`

Allows you to specify a particular version of an OSTree and deploy it. This command is more flexible than `upgrade` or `rollback`, as they only alternate between the two installed OSTrees. The newly downloaded tree replaces the one that is not currently deployed. The syntax is as follows:

```
atomic host deploy <version/commit ID>
```

For example, use this command to deploy the 7.2.1 OSTree and initiate a reboot after the tree is downloaded:
# atomic host deploy 7.2.1 -r

Use the `--preview` option to check the package difference between your currently deployed tree and a specified one:

```
# atomic host deploy 7.2.1 --preview
```

If you are unsure about the version numbering, pull the commit history for the repository you are subscribed to by using the following `ostree` commands:

```
# ostree pull --commit-metadata-only --depth -1 rhel-atomic-host-ostree:rhel-atomic-host/7/x86_64/standard
# ostree log rhel-host/7/x86_64/standard
```

When you have the version number you are interested in you can use the `atomic host <version> --preview` command to check the package differences.

You can have at most two deployments on the system. `upgrade` or `deploy` downloads a new tree and replaces the currently not deployed one. You can then alternate between both trees on the system with `rollback`.

You can also use the commit ID of a particular version. The following Solution from the Customer Portal contains a list of all commit IDs that have been released: Deploying a specific version of Red Hat Enterprise Linux Atomic Host.

### 3.2. ATOMIC DIFF

Compares two images or containers at a file level and displays a full list of their differences. By default, a full list of files is displayed.

```
atomic diff <image1> <image2>
```

You can modify the output with a combination of several options.

```
# atomic diff --rpms --no-files rhel7 centos
```

```
<table>
<thead>
<tr>
<th>rhel7</th>
<th>centos</th>
</tr>
</thead>
<tbody>
<tr>
<td>----------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td><code>Red Hat Enterprise Linux Server release 7.2 (Maipo)</code></td>
<td><code>CentOS Linux release 7.2.1511 (Core)</code></td>
</tr>
<tr>
<td>bind-license-32-9.9.4</td>
<td>centos-release-0-7</td>
</tr>
<tr>
<td>hostname-0-3.13</td>
<td>hostapd-0.5.1</td>
</tr>
<tr>
<td>iputils-0-20121221</td>
<td>python-etu-0-0.8</td>
</tr>
<tr>
<td>m2crypto-0-0.20.1</td>
<td>python-entitytool-0-0.8</td>
</tr>
<tr>
<td>python-dateutil-0-1.5</td>
<td>python-ethtool-0-0.8</td>
</tr>
</tbody>
</table>
```
The **--rpms** option adds a table with differences between the RPMs in the two images. Combined with the **--no-files** option you can restrict the output to only print that table.

---

**Warning**

Do not use the **--no-files** option on its own as it will not produce any output.

---

The **--names-only** option compares package names only, without versions.

It is a good idea to redirect the output to a text viewer such as **less**, as the full list of files can get too long and the terminal will truncate it.

```bash
# atomic diff rhel7 centos | less
```

Use the **--json** option to print the output in JSON format and redirect it, for example, with **less**:

```bash
# atomic diff --rpms --json rhel7 centos | less
```

### 3.3. ATOMIC INSTALL

```bash
atomic install <image>
```

Executes an image’s install method. The install method is described in the **LABEL INSTALL** field in the container image. It is typically used to prepare the host system to run the image. It often exposes configuration files needed for the image to the host so they can be edited and saved if the image is deleted. For example, this install method:

```bash
# atomic info rhel7/rsyslog
[output truncated]
INSTA: docker run --rm --privileged -v /:/host -e HOST=/host -e
IMAGE=IMAGE -e NAME=NAME IMAGE /bin/install.sh
```

executes the following command:

```bash
# atomic install rhel7/rsyslog
docker run --rm --privileged -v /:/host -e HOST=/host -e
IMAGE=rhel7/rsyslog -e NAME=rsyslog rhel7/rsyslog /bin/install.sh
```

With this instruction, **atomic install** mounts files from the root directory (/) on the host to the /host/ directory inside the container and sets the $HOST variable as /host/ inside the container. For example, `/usr/bin` is `/host/usr/bin` in the container, `$IMAGE` is `rhel7/rsyslog` and `$NAME` is `rsyslog`. The `/bin/install.sh` script exposes the `/etc/rsyslog.conf` file to the host system so you can edit it from outside the container.
If you do not have the image locally, `atomic install` pulls the image from a configured registry. Use the `--display` option to show the image's install method. The install command does not execute if `--display` is specified.

Use the `-n` option to install multiple copies of an image:

```
# atomic install -n name1 rhel7/rsyslog
# atomic install -n name2 rhel7/rsyslog
```

### 3.4. ATOMIC UNINSTALL

`atomic uninstall <image>`

Similar to `install`, `uninstall` reads and executes an image's uninstall method from the `UNINSTALL` instruction.

### 3.5. ATOMIC RUN

`atomic run <image>`

Executes an image's run method. The run method is described in the `RUN` field in the container image. `RUN` allows a developer to define how the particular application should be run. For example, a container with the `ntpd` service requires the `--cap_add SYS_TIME` option, and the option can be embedded into the `RUN` label instead of the user typing the following full command:

```
# docker run -d -n --cap_add SYS_TIME ntpd
```

If the `RUN` field does not exist, `atomic run` defaults to running the following:

```
docker create -ti -n <image_name> <container_name>
```

Use the `--spc` option to run a container in super-privileged mode. You can read more about Super-Privileged Containers here: Chapter 9. Running Super-privileged Containers from the RHEL Atomic Host Getting Started with Containers Guide.

### 3.6. ATOMIC STOP

`atomic stop <image_name>/<container_name>`

Executes an image's stop method. Use this command to stop running containers. Takes the image name or container name as argument. For example:

```
# atomic stop cranky_wright
```

or

```
# atomic stop rhel7/rsyslog
```

### 3.7. ATOMIC IMAGES
Executes commands on images. You can view your images, display LABEL info or their help file, check for newer versions

- **atomic images list**

Lists the container images you have downloaded on your system. The > symbol indicates that the image is being used by a container.

```
# atomic images list

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>registry.access.redhat.com/rhel7/openscap</td>
<td>latest</td>
</tr>
<tr>
<td>sha256:da0d5</td>
<td>2016-06-20 14:24</td>
</tr>
<tr>
<td>registry.access.redhat.com/rhel7/sadc</td>
<td>latest</td>
</tr>
<tr>
<td>sha256:7ed99</td>
<td>2016-05-08 16:31</td>
</tr>
<tr>
<td>registry.access.redhat.com/rhel7/kubernetes-controller-mgr</td>
<td>latest</td>
</tr>
<tr>
<td>sha256:feb3d</td>
<td>2016-05-06 20:12</td>
</tr>
<tr>
<td>registry.access.redhat.com/rhel7/kubernetes-apiserver</td>
<td>latest</td>
</tr>
<tr>
<td>sha256:c3ac0</td>
<td>2016-05-06 20:12</td>
</tr>
<tr>
<td>registry.access.redhat.com/rhel7/kubernetes-scheduler</td>
<td>latest</td>
</tr>
<tr>
<td>sha256:d6c72</td>
<td>2016-05-06 20:12</td>
</tr>
<tr>
<td>registry.access.redhat.com/rhel7/cockpit-ws</td>
<td>latest</td>
</tr>
<tr>
<td>sha256:f1ea2</td>
<td>2016-05-06 18:54</td>
</tr>
<tr>
<td>registry.access.redhat.com/rhel7/rhel-tools</td>
<td>latest</td>
</tr>
<tr>
<td>sha256:00211</td>
<td>2016-05-06 17:49</td>
</tr>
<tr>
<td>registry.access.redhat.com/rhel7/rsyslog</td>
<td>latest</td>
</tr>
<tr>
<td>sha256:92bd7</td>
<td>2016-05-06 17:40</td>
</tr>
</tbody>
</table>
```

- **atomic images delete**

Delete a specified image from your system. By default, you won't be able to delete an image which has containers based on it. Use the `-f` option to force remove that image. This will not stop the running container based on that image.

```
# atomic images delete -f rhel7/rsyslog
```

Use the `--remote` option to delete an image from a remote repository. However, the remote disk space will not be free until the `registry garbage-collection` command is run for the remote registry.

- **atomic images info**

Displays information about an image:

```
# atomic images info rhel7/rhel-tools:latest
Image Name: registry.access.redhat.com/rhel7/rhel-tools:latest
distribution-scope: public
build-date: 2016-09-09T14:41:51.833402Z
RUN: docker run -it --name NAME --privileged --ipc=host --net=host --pid=host -e HOST=/host -e NAME=NAME -e IMAGE=IMAGE -v /run:/run -v
```
By default, it checks in local images first and then tries the registries you have configured on your system. Use the `--remote` option to ignore the local images and look only in the configured registries:

```shell
# atomic images info --remote rhel7/rhel-tools
```

## atomic images prune

Use `atomic images prune` to free disk space by deleting unused *dangling* images. Dangling images are those with no name or tag and that are not used by any other images. Since they are not used, they occupy system space. Dangling images are usually caused by using the `docker build` command to update an image without also removing the older version of the image. An asterisk (*) indicates a dangling image:

```shell
# atomic images list -a
```

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IMAGE ID</td>
</tr>
<tr>
<td></td>
<td>registry.access.redhat.com/rhel7/openscap</td>
</tr>
<tr>
<td></td>
<td>sha256:da0d5</td>
</tr>
<tr>
<td>&gt;</td>
<td>registry.access.redhat.com/rhel7/sadc</td>
</tr>
<tr>
<td></td>
<td>sha256:7ed99</td>
</tr>
<tr>
<td>&gt;</td>
<td>registry.access.redhat.com/rhel7/kubernetes-controller-mgr</td>
</tr>
<tr>
<td></td>
<td>sha256:feb3d</td>
</tr>
<tr>
<td>&gt;</td>
<td>registry.access.redhat.com/rhel7/kubernetes-apiserver</td>
</tr>
<tr>
<td></td>
<td>sha256:c3ac0</td>
</tr>
<tr>
<td></td>
<td>registry.access.redhat.com/rhel7/kubernetes-scheduler</td>
</tr>
<tr>
<td></td>
<td>sha256:d6c72</td>
</tr>
<tr>
<td><em>&lt;none&gt;</em></td>
<td>sha256:bad41</td>
</tr>
<tr>
<td><em>&lt;none&gt;</em></td>
<td>sha256:9339b</td>
</tr>
<tr>
<td>&gt;</td>
<td>registry.access.redhat.com/rhel7/cockpit-ws</td>
</tr>
<tr>
<td></td>
<td>sha256:f1ea2</td>
</tr>
<tr>
<td></td>
<td>registry.access.redhat.com/rhel7/rhel-tools</td>
</tr>
<tr>
<td></td>
<td>sha256:00211</td>
</tr>
<tr>
<td>&gt;</td>
<td>registry.access.redhat.com/rhel7/rsyslog</td>
</tr>
<tr>
<td></td>
<td>sha256:92bd7</td>
</tr>
</tbody>
</table>
atomic images verify

atomic images verify <image>

Checks if there is a newer image available. It also scans through all layers to see if any of the sublayers have a new version available.

atomic images version

atomic images version <image>

Displays the "Name Version Release" label of an image.

```
# atomic version rhel7/rsyslog
00b31ffda5e92737fe07aeca972d6fb4bda7cc8eca225f6a12e06db1ac5ba39
rhel7/rsyslog-7.1-29 registry.access.redhat.com/rhel7/rsyslog:latest
```

3.8. ATOMIC CONTAINERS

Executes commands on containers. With this command and the subcommands you can list the currently running containers, delete or trim them.

atomic containers list

```
# atomic containers list
CONTAINER ID IMAGE                COMMAND              CREATED
STATUS    RUNTIME
flannel      rhel7/flannel        /usr/bin/flanneld-ru 2016-10-06
14:36 running   runc
etcd         rhel7/etcd           /usr/bin/etcd-env.sh 2016-10-13
14:21 running   runc
1cf730472572 rhel7/cockpit-ws     /container/atomic-ru 2016-10-13
17:55 running   Docker
```

Lists all running containers on the system with information about them, including which runtime a container is using, Docker, or runc (docker ps lists only the Docker-formatted containers).

atomic containers list -a will show all containers:

```
# atomic containers list -a
CONTAINER ID IMAGE                COMMAND              CREATED
STATUS    RUNTIME
etcd         rhel7/etcd           /usr/bin/etcd-env.sh 2016-10-13
14:21 running   runc
flannel      rhel7/flannel        /usr/bin/flanneld-ru 2016-10-13
15:12 failed   runc
1cf730472572 rhel7/cockpit-ws     /container/atomic-ru 2016-10-13
17:55 running   Docker
9a2bb24e5978 rhel7/rsyslog        /bin/rsyslog.sh      2016-10-13
17:49 created   Docker
34f95af8f8f9 rhel7/cockpit-ws     /container/atomic-ru 2016-09-27
19:10 exited   Docker
```

atomic containers list also supports filtering the output with the -f option. The filters are: container ID, image, command, created, status, runtime. For example:
# atomic containers list -f status=exited

1cf730472572 rhel7/cockpit-ws /container/atomic-ru 2016-10-13 17:55 exited Docker
34f95af8f8f9 rhel7/cockpit-ws /container/atomic-ru 2016-09-27 19:10 exited Docker

atomic containers delete

Deletes a specifies container, for example:

# atomic containers delete rhel7/flannel

atomic containers trim

This command discards unused blocks from running containers. It uses the fstrim command that discards blocks which are not used by the file system. It is especially useful for Thinly-Provisioned storage which is the option used on RHEL Atomic Host. Use this command about once a week to clean up the system from unused file system blocks. For more detailed information, see the fstrim(8) manual page.

3.9. ATOMIC TOP

atomic top [<container>]

Displays an interactive view of the processes running in active containers, like the top utility. By default, atomic top monitors all containers, but you can optionally specify only the containers you want by using the container name or ID. The table with default fields looks like this:

<table>
<thead>
<tr>
<th>CONTAINER</th>
<th>NAME</th>
<th>PID</th>
<th>CPU</th>
<th>MEM</th>
<th>UID</th>
<th>GID</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec56d2f1fb10</td>
<td>httpd</td>
<td>2087</td>
<td>0.0</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>httpd -</td>
</tr>
<tr>
<td>DFOREGROUND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ec56d2f1fb10</td>
<td>httpd</td>
<td>2095</td>
<td>0.0</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>httpd -</td>
</tr>
<tr>
<td>DFOREGROUND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ec56d2f1fb10</td>
<td>httpd</td>
<td>2096</td>
<td>0.0</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>httpd -</td>
</tr>
<tr>
<td>DFOREGROUND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ec56d2f1fb10</td>
<td>httpd</td>
<td>2097</td>
<td>0.0</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>httpd -</td>
</tr>
<tr>
<td>DFOREGROUND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fa7586391e42</td>
<td>fedora</td>
<td>1913</td>
<td>0.0</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>/bin/sh</td>
</tr>
</tbody>
</table>

You can sort the processes by pressing the key in the parenthesis from the column headers. For example, press "P" to sort processes by PID.

# atomic top -d 5 -n 3

With this command, you can monitor processes on a five second interval for three iterations.

To add add more fields to the default ones, use the --optional option, for example parent PIDs and UID:

# atomic top --optional ppid uid
3.10. ATOMIC MOUNT

atomic mount <image> <mountpoint>

Mounts the underlying file system of a container or image into the host file system. This way you can inspect their contents. For example, you can use it to check configuration files.

Accepts one of image UUID, container UUID, container NAME, or image REPO (optionally with registry and tag information). If the given UUID or NAME is a container, and the --live option is not set, then atomic mount creates a snapshot of the container by committing it to a temporary image and spawning a temporary container from that image. If UUID or REPO refers to an image, then atomic mount creates a temporary container from the given image. All temporary artifacts are cleaned upon unmounting.

```
# mkdir /root/tmp
# atomic mount rhel7/rsyslog /root/tmp
# cd /root/tmp
# ls
```

**Note**

atomic mount is only supported on the devicemapper and overlayfs storage backends.

3.11. ATOMIC UNMOUNT

atomic unmount <mountpoint>

Unmounts a container or image previously mounted with atomic mount.

```
# atomic unmount /root/tmp
```

3.12. ATOMIC PULL

atomic pull <image>

Fetches an image from a repository and downloads it to the system:

```
# atomic pull rhel7/rsyslog
```

You can also specify the source using the source:image format. These are the following options for source:

- **oci**: fetches an image from a Docker registry using the skopeo tool. This is the default option that is assumed when no source is specified. You can change the default by editing the /etc/atomic.conf file with the default_storage keyword.

```
# atomic pull oci:rhel7/etcd
```
- **docker**: imports an image from a local Docker registry, not accessing the network. It is equivalent to saving the image from docker (docker save IMAGE) and importing it into the OSTree repository:

```bash
# atomic pull --storage=ostree docker:fedora
```

- **dockertar**: imports a tarball from a local Docker registry

```bash
# atomic pull --storage=ostree dockertar:path/to/image.tar
```

- **ostree**: fetches an image from a remote OSTree repository. The remote has to be already configured in the local OSTree repository:

```bash
# atomic pull --storage=ostree ostree:<remote>/branch
```

Use the --storage option to specify a destination storage for the image. The two options are docker and ostree. If unspecified, the command assumes it is docker. Use the ostree option when pulling system container images:

```bash
# atomic pull --storage=ostree rhel7/etcd
```

Use the --type option to specify a different registry type. You can switch to an atomic type of registry. For example:

```bash
# atomic pull --type atomic <atomic_registry_address>:namespace/image
```

### 3.13. ATOMIC PULL

```bash
atomic pull <new_image>
```

Pushes an image you have built locally to a repository. The default behavior is to push to a docker repository, but can also be set to push to a Satellite or Pulp repository with the --satellite or --pulp options.

### 3.14. ATOMIC STORAGE (MIGRATE)

Manages container storage.

- **atomic storage export/import**

With the export and import commands, you can migrate all images, volumes, and containers from one version of atomic to another, or from one storage backend to another. With atomic export you can save all data from the current atomic instance, change the environment, and then import all their old data to the new system with atomic import. This command was previously called "migrate".

```bash
# atomic storage export
```

Will export all current images, volumes, and containers to /var/lib/atomic/migrate/ (or another specified directory), under the /images/, /volumes/, and /containers/ subdirectories.

```bash
# atomic storage import
```
Will import the images, volumes, and containers previously saved in `/var/lib/atomic/migrate/`, or another specified directory into the new atomic instance.

If you are running docker from a custom location (not `/var/lib/docker/`), you must set the `--graph` option pointing to the custom location. To save the data in a non-standard directory, use the `--dir` option.

- **atomic storage modify**

Modifies the default storage setup.

You can add a block device to the storage pool. This command expands your devicemapper storage pool by adding the block device. Only works with devicemapper driver. For example:

```bash
# atomic storage modify --add-device vda3/rhelah-expand
```

To change the backend storage driver, use the `--driver` option. The supported drivers are `devicemapper` and `overlay`.

```bash
# atomic storage modify --driver overlay
```

- **atomic storage reset**

This command deletes all containers and images from your system and resets the storage settings to their default values.

### 3.15. ATOMIC SCAN

**atomic scan <image>/</container>**

Scans images and containers for Common Vulnerabilities and Exposures (CVEs). By default, **atomic scan** uses the `openscap` scanner to scan the images, but the pluggable design supports adding more scanners, including custom ones. When you run **atomic scan** the first time, it downloads the `rhel7/openscap` container which provides the `openscap` scanner. The default scan type for `openscap` is to check for vulnerabilities. Note that `openscap` works with RHEL-based images and containers only.

For example, to scan the `rhel7` base image, run:

```bash
# atomic scan rhel7/rhel
```

To scan all containers and images and produce a detailed report, run:

```bash
# atomic scan --all --verbose
```

If the results are positive, the output is similar to the following:

```bash
# atomic scan rhel7/rhel
```
rhel7/rhel (sha256:bf203442)

The following issues were found:

RHSA-2016:1025: pcre security update (Important)
Severity: Important
RHSA URL: https://rhn.redhat.com/errata/RHSA-2016-1025.html
RHSA ID: RHSA-2016:1025-00
Associated CVEs:
CVE ID: CVE-2015-2328
CVE URL: https://access.redhat.com/security/cve/CVE-2015-2328
CVE ID: CVE-2015-3217
CVE URL: https://access.redhat.com/security/cve/CVE-2015-3217
CVE ID: CVE-2015-5073
CVE URL: https://access.redhat.com/security/cve/CVE-2015-5073
CVE ID: CVE-2015-8385
CVE URL: https://access.redhat.com/security/cve/CVE-2015-8385
CVE ID: CVE-2015-8386
CVE URL: https://access.redhat.com/security/cve/CVE-2015-8386
CVE ID: CVE-2015-8388
CVE URL: https://access.redhat.com/security/cve/CVE-2015-8388
CVE ID: CVE-2015-8391
CVE URL: https://access.redhat.com/security/cve/CVE-2015-8391
CVE ID: CVE-2016-3191
CVE URL: https://access.redhat.com/security/cve/CVE-2016-3191

Files associated with this scan are in /var/lib/atomic/openscap/2016-06-21-10-10-28-942890.

To list all configured scanners, use:

```bash
# atomic scan --list
Scanner: openscap *
  Image Name: rhel7/openscap
  Scan type: cve *
    Description: Performs a CVE scan based on known CVE data
  Scan type: standards_compliance
    Description: Performs a standards scan

* denotes defaults
```

The output also lets you check what scan types are available for each scanner. openscap has two defined, and you can use the --scan_type option to switch between both:

```bash
# atomic scan --scan_type standards_compliance rhel7/rhel
docker run -it --rm -v /etc/localtime:/etc/localtime -v
/run/atomic/2016-07-12-16-08-03-011887:/scanin -v
/var/lib/atomic/openscap/2016-07-12-16-08-03-011887:/scanout:rw,Z -v
/etc/oscadp:/etc/oscadp:ro rhel7/openscap oscadp-evaluate scan --targets chroots-in-dir:////scanin --output /scanout --no-cve-scan
```

rhel7 (sha256:5fbb7430)

The following issues were found:
Ensure Software Patches Installed
Severity: Important
XCCDF result: notchecked

Files associated with this scan are in /var/lib/atomic/openscap/2016-07-12-16-08-03-011887.

Adding a new scanner means simply installing a new image that provides that scanner with atomic install, and if it is a custom one that you have locally, use:

```bash
# atomic install localhost:5000/custom_scanner
```

You can use the new scanner with the --scanner option:

```bash
# atomic scan --scanner custom_scanner rhel7/rhel
```

To change the default scanner, edit the default_scanner line in /etc/atomic.conf. You can also use this line to explicitly set openscap as the default. If it is not set explicitly, atomic scan uses openscap.

```bash
default_scanner: custom_scanner
```

Another feature of atomic scan is that it can also scan the host file system. This can be configured using the --rootfs option and providing a path on the host, for example:

```bash
# atomic scan --rootfs /tmp/chroot
```

### 3.16. ATOMIC SIGN

```bash
# atomic sign <registry>/<image>
```

Creates a local signature for one or more local images that have been pulled from a registry. By default, the signature is written into a directory derived from the registry configuration files as configured in the /etc/atomic.conf file using the registry_confdir keyword.

**Warning**

Only use atomic sign if you trust the remote registry which contains the image. It is recommended that this is a registry which you administer.

Use the -d option to save the signature in a different than the default location:

```bash
# atomic sign -d /tmp/signatures myregistry.example.com/my_image
```

Use the --sign-by option to the default identity specified in the /etc/atomic.conf file and use --gnupghome to provide a location to that identity's keyring.

```bash
# atomic sign --sign-by user@example.com --gnupghome=~/.gnupg
```
myregistry.example.com/my_image

For detailed information about image signing, see Signing Container Images chapter from the Red Hat Enterprise Linux Atomic Host Managing Containers Guide.

### 3.17. ATOMIC TRUST

The **atomic trust** command manages the trust policy of the host system. The trust policy is stored in the `/etc/containers/policy.json` file and defines a scope of registries or repositories or both that must be signed by public keys. Trust is enforced when a user attempts to pull an image from a registry.

- **atomic trust show**

  Displays the contents of the `/etc/containers/policy.json` file:
  ```
  # atomic trust show
  * (default)                         accept
  
  # atomic trust default
  
  Manages the default trust policy. Use the **accept** or **reject** commands to enable or disable the default trust policy.
  ```

  ```
  # atomic trust default reject
  
  or
  ```

  ```
  # atomic trust default accept
  
  # atomic trust add
  
  Updates the trust policy. To add a public key, use:
  ```
  # atomic trust add  --pubkeys /etc/pki/containers/foo@example.com --sigstore https://server.example.com/foobar/sigstore/ <registry>/<image>
  
  To accept all unsigned images from a registry:
  ```
  # atomic trust add --type insecureAcceptAnything <registry>
  
  # atomic trust delete
  
  Removes a trust scope. For example:
  ```
  # atomic trust delete <registry>
  
  For detailed information about image signing, see Signing Container Images chapter from the Red Hat Enterprise Linux Atomic Host Managing Containers Guide.

### 3.18. ATOMIC UPDATE

- **atomic update <image>**
Pulls the latest update of an image from the configured repositories. If a container based on this image exists, the container will continue to use the old image. Use the \texttt{--force} option to remove the container. An example output:

```
# atomic update rhel7/rsyslog
Using default tag: latest
00b31ffda5e9: Download complete
c4f590bbce3: Download complete
Status: Image is up to date for
registry.access.redhat.com/rhel7/rsyslog:latest
```

### 3.19. ATOMIC --HELP AND MANUAL PAGES

The \texttt{--help} option is available to \texttt{atomic} and all of the atomic subcommands described in this document. Use \texttt{--help} to print a usage message and all of the available options to a subcommand.

As RHEL Atomic Host does not have man pages on the OSTree, you can access the manual pages for \texttt{atomic} and \texttt{rpm-ostree} through the \texttt{Red Hat Enterprise Atomic Tools} container. Use the following commands:

```
# atomic install rhel7/rhel-tools
# atomic run rhel7/rhel-tools man atomic
```

Individual commands are hyphenated, so use the following format:

```
# atomic run rhel7/rhel-tools man atomic-mount
```

You can access the \texttt{rpm-ostree} manual pages using the same commands.