Rocky Enterprise Linux 9.2 Manual Pages on command 'podman-system-events.1'

$ man podman-system-events.1

podman-events(1) General Commands Manual podman-events(1)

NAME

podman-events - Monitor Podman events

SYNOPSIS

podman events [options]

podman system events [options]

DESCRIPTION

Monitor and print events that occur in Podman. Each event will include a timestamp, a type, a status, name (if applicable), and image (if ap? plicable). The default logging mechanism is journald. This can be changed in containers.conf by changing the events_logger value to file. Only file and journald are accepted. A none logger is also available but this logging mechanism completely disables events; nothing will be reported by podman events.

By default, streaming mode is used, printing new events as they occur. Previous events can be listed via --since and --until.

The container event type will report the follow statuses:

* attach

* checkpoint
* cleanup
* commit
* connect
* create
* died
* disconnect
* exec
* exec_died
* exited
* export
* import
* init
* kill
* mount
* pause
* prune
* remove
* rename
* restart
* restore
* start
* stop
* sync
* unmount
* unpause

The pod event type will report the follow statuses:

* create
* kill
* pause
* remove
* start
* stop

* unpause

The image event type will report the following statuses:

* loadFromArchive,

* mount

* pull

* push

* remove

* save

* tag

* unmount

* untag

The system type will report the following statuses:

* refresh

* renumber

The volume type will report the following statuses:

* create

* prune

* remove

Verbose Create Events

Setting events_container_create_inspect_data=true in containers.conf(5) instructs Podman to create more verbose container-create events which include a JSON payload with detailed information about the containers. The JSON payload is identical to the one of podman-container-in?spect(1). The associated field in journald is named PODMAN_CON?TAINER_INSPECT_DATA.

OPTIONS

--filter, -f=filter

Filter events that are displayed. They must be in the format of "fil?ter=value". The following filters are supported:

* container=name_or_id

* event=event_status (described above)

* image=name_or_id

* label=key=value

* pod=name_or_id

* volume=name_or_id

* type=event_type (described above)

In the case where an ID is used, the ID may be in its full or shortened form. The "die" event is mapped to "died" for Docker compatibility.

--format

Format the output to JSON Lines or using the given Go template.

??

?Placeholder ? Description ?

??

?.Attributes ? created_at, _by, labels, and ?

? ? more (map[]) ?

??

?.ContainerExitCode ? Exit code (int) ?

??

?.Details ... ? Internal structure, not actu? ?

? ? ally useful ?

??

?.HealthStatus ? Health Status (string) ?

??

?.ID ? Container ID (full 64-bit SHA) ?

??

?.Image ? Name of image being run ?

? ? (string) ?

??

?.Name ? Container name (string) ?

??

?.Network ? Name of network being used ?

? ? (string) ?

??

?.Status ? Event status (e.g., create, ?

? ? start, died, ...) ?

??

?.Time ? Event timestamp (string) ?

??

?.Type ? Event type (e.g., image, con? ?

? ? tainer, pod, ...) ?

??

--help

Print usage statement.

--no-trunc

Do not truncate the output (default true).

--since=timestamp

Show all events created since the given timestamp

--until=timestamp

Show all events created until the given timestamp

The since and until values can be RFC3339Nano time stamps or a Go dura?

tion string such as 10m, 5h. If no since or until values are provided,

only new events will be shown.

JOURNALD IDENTIFIERS

The journald events-backend of Podman uses the following journald iden?

tifiers. You can use the identifiers to filter Podman events directly

with journalctl.

??

?Identifier ? Description ?

??

?SYSLOG_IDENTIFIER ? Always set to "podman" ?

??

?PODMAN_EVENT ? The event status as described above ?

??

?PODMAN_TYPE ? The event type as described above ?

??

?PODMAN_TIME ? The time stamp when the event was written ?

??

?PODMAN_NAME ? Name of the event object (e.g., container, image) ?

??

?PODMAN_ID ? ID of the event object (e.g., container, image) ?

??

?PODMAN_EXIT_CODE ? Exit code of the container ?

??

?PODMAN_POD_ID ? Pod ID of the container ?

??

?PODMAN_LABELS ? Labels of the container ?

??

?PODMAN_HEALTH_STATUS ? Health status of the container ?

??

?PODMAN_CONTAINER_INSPECT_DATA ? The JSON payload of podman-inspect as described above ?

??

?PODMAN_NETWORK_NAME ? The name of the network ?

??

EXAMPLES

Showing Podman events

$ podman events

2019-03-02 10:33:42.312377447 -0600 CST container create 34503c192940 (image=docker.io/library/alpine:latest, name=friendly_allen)

2019-03-02 10:33:46.958768077 -0600 CST container init 34503c192940 (image=docker.io/library/alpine:latest, name=friendly_allen)

2019-03-02 10:33:46.973661968 -0600 CST container start 34503c192940 (image=docker.io/library/alpine:latest, name=friendly_allen)

2019-03-02 10:33:50.833761479 -0600 CST container stop 34503c192940 (image=docker.io/library/alpine:latest, name=friendly_allen)

2019-03-02 10:33:51.047104966 -0600 CST container cleanup 34503c192940 (image=docker.io/library/alpine:latest, name=friendly_allen)

Show only Podman create events

$ podman events -f event=create

2019-03-02 10:36:01.375685062 -0600 CST container create 20dc581f6fbf (image=docker.io/library/alpine:latest, name=sharp_morse)

2019-03-02 10:36:08.561188337 -0600 CST container create 58e7e002344c (image=k8s.gcr.io/pause:3.1, name=3e701f270d54-infra)

2019-03-02 10:36:13.146899437 -0600 CST volume create cad6dc50e087 (image=, name=cad6dc50e0879568e7d656bd004bd343d6035e7fc4024e1711506fe2fd459e6f)

2019-03-02 10:36:29.978806894 -0600 CST container create d81e30f1310f (image=docker.io/library/busybox:latest, name=musing_newton)

Show only Podman pod create events

$ podman events --filter event=create --filter type=pod

2019-03-02 10:44:29.601746633 -0600 CST pod create 1df5ebca7b44 (image=, name=confident_hawking)

2019-03-02 10:44:42.374637304 -0600 CST pod create ca731231718e (image=, name=webapp)

2019-03-02 10:44:47.486759133 -0600 CST pod create 71e807fc3a8e (image=, name=reverent_swanson)

Show only Podman events created in the last five minutes:

$ sudo podman events --since 5m

2019-03-02 10:44:29.598835409 -0600 CST container create b629d10d3831 (image=k8s.gcr.io/pause:3.1, name=1df5ebca7b44-infra)

2019-03-02 10:44:29.601746633 -0600 CST pod create 1df5ebca7b44 (image=, name=confident_hawking)

2019-03-02 10:44:42.371100253 -0600 CST container create 170a0f457d00 (image=k8s.gcr.io/pause:3.1, name=ca731231718e-infra)

2019-03-02 10:44:42.374637304 -0600 CST pod create ca731231718e (image=, name=webapp)

Show Podman events in JSON Lines format

$ podman events --format json

{"ID":"683b0909d556a9c02fa8cd2b61c3531a965db42158627622d1a67b391964d519","Image":"localhost/myshdemo:latest","Name":"agitated_diffie","Status":"cleanup","Time":"2019-04-27T22:47:00.849932843-04:00","Type":"container"}

{"ID":"a0f8ab051bfd43f9c5141a8a2502139707e4b38d98ac0872e57c5315381e88ad","Image":"docker.io/library/alpine:latest","Name":"friendly_tereshkova","Status":"unmount","Time":"2019-04-28T13:43:38.063017276-04:00","Type":"container"}

SEE ALSO

podman(1), containers.conf(5)

HISTORY

March 2019, Originally compiled by Brent Baude bbaude@redhat.com

?mailto:bbaude@redhat.com?