



Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'docker-pod-create.1'

\$ man docker-pod-create.1

podman-pod-create(1()) podman-pod-create(1())

NAME

podman-pod-create - Create a new pod

SYNOPSIS

podman pod create [options]

DESCRIPTION

Creates an empty pod, or unit of multiple containers, and prepares it to have containers added to it. The pod id is printed to STDOUT. You can then use `podman create --pod <pod_id|pod_name> ...` to add containers to the pod, and `podman pod start <pod_id|pod_name>` to start the pod.

OPTIONS

`--add-host=host:ip`

Add a host to the `/etc/hosts` file shared between all containers in the pod.

`--cgroup-parent=path`

Path to cgroups under which the cgroup for the pod will be created. If the path is not absolute, the path is considered to be relative to the cgroups path of the init process.

Cgroups will be created if they do not already exist.

`--dns=ipaddr`

Set custom DNS servers in the `/etc/resolv.conf` file that will be shared between all containers in the pod. A special option, "none" is allowed which disables creation of `/etc/resolv.conf` for the pod.

`--dns-opt=option`

Set custom DNS options in the `/etc/resolv.conf` file that will be shared between all containers in the pod.

ainers in the pod.

`--dns-search=domain`

Set custom DNS search domains in the `/etc/resolv.conf` file that will be shared between all containers in the pod.

`--gidmap=container_gid:host_gid:amount`

GID map for the user namespace. Using this flag will run the container with user namespace enabled. It conflicts with the `--users` and `--subgidname` flags.

`--uidmap=container_uid:from_uid:amount`

Run the container in a new user namespace using the supplied mapping. This option conflicts with the `--users` and `--subuidname` options. This option provides a way to map host UIDs to container UIDs. It can be passed several times to map different ranges.

`--subgidname=name`

Name for GID map from the `/etc/subgid` file. Using this flag will run the container with user namespace enabled. This flag conflicts with `--users` and `--gidmap`.

`--subuidname=name`

Name for UID map from the `/etc/subuid` file. Using this flag will run the container with user namespace enabled. This flag conflicts with `--users` and `--uidmap`.

`--help, -h`

Print usage statement.

`--hostname=name`

Set a hostname to the pod

`--infra=true|false`

Create an `infra` container and associate it with the pod. An `infra` container is a lightweight container used to coordinate the shared kernel namespace of a pod. Default: `true`.

`--infra-common-pidfile=file`

Write the pid of the `infra` container's common process to a file. As `common` runs in a separate process than `Podman`, this is necessary when using `systemd` to manage `Podman` containers and pods.

`--infra-command=command`

The command that will be run to start the `infra` container. Default: `"/pause"`.

`--infra-image=image`

The image that will be created for the `infra` container. Default: `"k8s.gcr.io/pause:3.1"`.

`--infra-name=name`

The name that will be used for the pod's infra container.

`--ip=ipaddr`

Set a static IP for the pod's shared network.

`--label=label, -l`

Add metadata to a pod (e.g., `--label com.example.key=value`).

`--label-file=label`

Read in a line delimited file of labels.

`--mac-address=address`

Set a static MAC address for the pod's shared network.

`--name=name, -n`

Assign a name to the pod.

`--network=mode, --net`

Set network mode for the pod. Supported values are: - bridge: Create a network stack on the default bridge. This is the default for rootfull containers. - none: Create a network namespace for the container but do not configure network interfaces for it, thus the container has no network connectivity. - host: Do not create a network namespace, all containers in the pod will use the host's network. Note: the host mode gives the container full access to local system services such as D-bus and is therefore considered insecure.

- network: Connect to a user-defined network, multiple networks should be comma-separated.

- private: Create a new namespace for the container. This will use the bridge mode for rootfull containers and slirp4netns for rootless ones. - slirp4netns[:OPTIONS,...]: use slirp4netns(1) to create a user network stack. This is the default for rootless containers. It is possible to specify these additional options:

- allow_host_loopback=true|false: Allow the slirp4netns to reach the host loopback IP (10.0.2.2, which is added to /etc/hosts as host.containers.internal for your convenience). Default is false.

- mtu=MTU: Specify the MTU to use for this network. (Default is 65520).

- cidr=CIDR: Specify ip range to use for this network. (Default is 10.0.2.0/24).

- enable_ipv6=true|false: Enable IPv6. Default is false. (Required for outbound_addr6).

- outbound_addr=INTERFACE: Specify the outbound interface slirp should bind to (ipv4 traffic only).

- outbound_addr=IPv4: Specify the outbound ipv4 address slirp should bind to.

- outbound_addr6=INTERFACE: Specify the outbound interface slirp should bind to (ipv6

traffic only).

- `outbound_addr6=IPv6`: Specify the outbound ipv6 address slirp should bind to.

- `port_handler=rootlesskit`: Use rootlesskit for port forwarding. Default.

Note: Rootlesskit changes the source IP address of incoming packets to a IP address in the container network namespace, usually 10.0.2.100. If your application requires the real source IP address, e.g. web server logs, use the `slirp4netns` port handler. The rootlesskit port handler is also used for rootless containers when connected to user-defined networks.

- `port_handler=slirp4netns`: Use the `slirp4netns` port forwarding, it is slower than root?

lesskit but preserves the correct source IP address. This port handler cannot be used for user-defined networks.

`--network-alias=strings`

Add a DNS alias for the pod. When the pod is joined to a CNI network with support for the `dnsname` plugin, the containers inside the pod will be accessible through this name from other containers in the network.

`--no-hosts=true|false`

Disable creation of `/etc/hosts` for the pod.

`--pid=pid`

Set the PID mode for the pod. The default is to create a private PID namespace for the pod. Requires the PID namespace to be shared via `--share`.

- `host`: use the host's PID namespace for the pod

- `ns`: join the specified PID namespace

- `private`: create a new namespace for the pod (default)

`--pod-id-file=path`

Write the pod ID to the file.

`--publish=port, -p`

Publish a port or range of ports from the pod to the host.

Format: `ip:hostPort:containerPort | ip::containerPort | hostPort:containerPort | containerPort`

Both `hostPort` and `containerPort` can be specified as a range of ports. When specifying ranges for both, the number of container ports in the range must match the number of

host ports in the range. Use `podman port` to see the actual mapping: `podman port CONTAINER $CONTAINERPORT`.

NOTE: This cannot be modified once the pod is created.

`--replace=true|false`

If another pod with the same name already exists, replace and remove it. The default is false.

--share=namespace

A comma-separated list of kernel namespaces to share. If none or "" is specified, no namespaces will be shared. The namespaces to choose from are ipc, net, pid, uts.

The operator can identify a pod in three ways: UUID long identifier

(?f78375b1c487e03c9438c729345e54db9d20cfa2ac1fc3494b6eb60872e74778?) UUID short identifier

(?f78375b1c487?) Name (?jonah?)

podman generates a UUID for each pod, and if a name is not assigned to the container with

--name then a random string name will be generated for it. The name is useful any place you need to identify a pod.

--users=mode

Set the user namespace mode for all the containers in a pod. It defaults to the POD?

MAN_USERNS environment variable. An empty value ("") means user namespaces are disabled.

Valid mode values are:

? auto[:OPTIONS,...]: automatically create a namespace. It is possible to specify these options to auto:

? gidmapping=_CONTAINER_GID:HOSTGID:SIZE to force a GID mapping to be present in the user namespace.

? size=SIZE: to specify an explicit size for the automatic user namespace. e.g.

--users=auto:size=8192. If size is not specified, auto will estimate a size for the user namespace.

? uidmapping=_CONTAINER_UID:HOSTUID:SIZE to force a UID mapping to be present in the user namespace.

? host: run in the user namespace of the caller. The processes running in the con?

tainer will have the same privileges on the host as any other process launched by the calling user (default).

? keep-id: creates a user namespace where the current rootless user's UID:GID are mapped to the same values in the container. This option is ignored for containers created by the root user.

EXAMPLES

```
$ podman pod create --name test
```

```
$ podman pod create --infra=false
```

```
$ podman pod create --infra-command /top
```

```
$ podman pod create --publish 8443:443
```

```
$ podman pod create --network slirp4netns:outbound_addr=127.0.0.1,allow_host_loopback=true
```

```
$ podman pod create --network slirp4netns:cidr=192.168.0.0/24
```

SEE ALSO

podman-pod(1)

HISTORY

July 2018, Originally compiled by Peter Hunt pehunt@redhat.com ?mailto:pehunt@redhat.com?

podman-pod-create(1)()