Using dcos task exec PREVIEW

The `dcos task exec` command allows you to execute an arbitrary command inside task's container and stream its output back to your local terminal to learn more about h given task is behaving. It offers an experience very similar to `docker exec`, without for SSH keys.

To use the debugging feature, the service or job must be launched using either the Mes container runtime or the Universal container runtime. Debugging cannot be used on con launched with the Docker runtime. See Using Mesos Containerizers for more informatio

You can execute this command in the following modes.

- `dcos task exec <task-id> <command>` (no flags): streams STDOUT and ST from the remote terminal to your local terminal as raw bytes.
- `dcos task exec --tty <task-id> <command>`: streams STDOUT and STD the remote terminal to your local terminal, but not as raw bytes. Instead, this option local terminal into raw mode, allocates a remote pseudo terminal (PYT), and streams STDOUT and STDERR through the remote PTY.
- `dcos task exec --interactive <task-id> <command>` streams STDOU STDERR from the remote terminal to your local terminal and streams STDIN from yo terminal to the remote command.
- `dcos task exec --interactive --tty <task-id> <command>`: streams STDOUT and STDERR from the remote terminal to your local terminal and streams S from your local terminal to the remote terminal. Also puts your local terminal into ra allocates a remote pseudo terminal (PYT); and streams STDOUT, STDERR, and STDI the remote PTY. This mode offers the maximum functionality.

Tips:

- We have included the text of the full flags above for readability, but each one can be shortened. Instead of typing `--interactive`, you can just type `-i`. Likewise, in typing `--tty`, you can just type `-t`.
- If your mode streams raw bytes, you won't be able to launch programs like `vim`, be these programs require the use of control characters.

For more information, see: the Command reference.

Quick start

Use this guide to get started with the `dcos task exec` debugging command.

Prerequisite:

- A container launched by using the DC/OS Universal container runtime.

Pipe output from a command running ins a container

You can run commands inside a container by using the `dcos task exec` command example, a long running Marathon app is launched and then the `dcos task exec` co is used to get the hostname of the node running the app.

1. Create a Marathon app definition and name it `my-app.json` with the following co

```
{
   "id": "/my-app",
   "cmd": "sleep 100000000",
   "cpus": 1,
   "instances": 1
 }
```

2. Deploy the service on DC/OS:

```
dcos marathon app add my-app.json
```

3. Get the task ID of the job with this CLI command:

```
dcos task
```

The output should look similar to this:

```
NAME        HOST         USER  STATE  ID
my-app      10.0.1.106   root    R    <task_id>
```

4. Run this command to show the hostname of the container running your app, where `ID>` is your task ID.

```
dcos task exec <task_id> hostname
```

The output should look similar to this:

```
ip-10-0-1-105.us-west-2.compute.internal
```

For more information about the `dcos task exec` command, see the CLI command reference.

Run an interactive command inside a tas container

You can run interactive commands on machines in your cluster by using the `dcos tas exec` command. In this example, the `dcos task exec` command is used to copy a script from your local machine to the task container on the node. The script is then adm locally by using the `dcos task exec` command.

1. Create a Marathon app definition and name it `my-interactive-app.json` with following contents:

```
{
   "id": "/my-interactive-app",
   "cmd": "sleep 100000000",
   "cpus": 1,
   "instances": 1
}
```

2. Deploy the app on DC/OS:

```
dcos marathon app add my-interactive-app.json
```

3. Get the task ID of the app with this CLI command:

```
dcos task
```

The output should look similar to this:

```
NAME                HOST        USER  STATE  ID
my-interactive-app  10.0.1.106  root    R    <task_id>
```

4. Write a script called `hello-world.sh` with the following contents:

```
echo "Hello World"
```

5. Upload the script to your task container:

```
cat hello-world.sh | dcos task exec -i <task_id> bash -c "cat > hello-world
```

6. Give the file executable permissions:

```
dcos task exec <task_id> chmod a+x hello-world.sh
```

7. Run the script inside of the container:

```
dcos task exec <task_id> ./hello-world.sh
```

The output should look similar to this:

```
Hello World
```

Launch a long running interactive Bash session

In this example, a long running job is launched by using the `dcos job run` comman `dcos task exec` command is used to launch an interactive Bash shell inside the cc of that job.

1. Deploy and run a job with the DC/OS CLI:
    1. Create the following job definition and save as `my-job.json`. This specifies a that runs for `10000000` seconds.

```
{
  "id": "my-job",
```

```
    "labels": {},
    "run": {
      "artifacts": [],
      "cmd": "sleep 100000000",
      "cpus": 0.01,
      "disk": 0,
      "env": {},
      "maxLaunchDelay": 3600,
      "mem": 32,
      "placement": {
        "constraints": []
      },
      "restart": {
        "policy": "NEVER"
      },
      "volumes": []
    }
  }
```

2. Deploy the job with this CLI command:

```
dcos job add my-job.json
```

3. Verify that the job has been successfully deployed:

```
dcos job list
```

   The output should resemble:

```
ID      DESCRIPTION  STATUS       LAST SUCCESFUL RUN
my-job               Unscheduled         None
```

4. Run the job:

```
dcos job run my-job
```

2. Get the task ID of the job with this CLI command:

```
dcos task
```

The output should look similar to this:

```
NAME                          HOST       USER  STATE  ID
20161209183121nz2F5.my-job    10.0.2.53  root    R    <task_id>
```

3. Launch a process inside of the container with the task ID (`<task_id>`) specified a attach a TTY to it. This will launch an interactive Bash session.

```
dcos task exec --interactive --tty <task_id> bash
```

You should now be inside the container running an interactive Bash session.

```
root@ip-10-0-2-53 / #
```

4. Run a command from the interactive Bash session. For example, the `ls` command

```
root@ip-10-0-1-104 / # ls
bin   dev  home  lib64       media  opt   root  sbin  sys  usr
boot  etc  lib   lost+found  mnt    proc  run   srv   tmp  var
```

Tip: You can use shorthand abbreviations `-i` for `--interactive` or `-t` for `--t`
Also, only the beginning unique characters of the `<task_id>` are required. For examp
task ID is `exec-test_20161214195` and there are no other task IDs that begin with
`e`, this is valid command syntax: `dcos task exec -i -t e bash`. For more info
see the CLI command reference.