

---

Subject: [PATCH 04/11] restart: improve success/failure reporting  
Posted by [Oren Laadan](#) on Mon, 07 Feb 2011 17:21:25 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

This patch improves the final status report (success/fail) of restart by adding a new ctx->success variable. The motivation is twofold:

- 1) Before, there was a confusion between who reports an error when (in --pidns) the root task isn't container init - in which case we add a dummy init. For this, we also improve how the dummy init communicates the status (success/fail) to the original restart task.
- 2) Even after restart succeeds, there may be errors (which may not affect the restarted task, but should be nevertheless reported).

Signed-off-by: Oren Laadan <[orenl@cs.columbia.edu](mailto:orenl@cs.columbia.edu)>

---

```
restart.c | 81 ++++++-----  
1 files changed, 46 insertions(+), 35 deletions(-)
```

```
diff --git a/restart.c b/restart.c  
index 9535543..a1af631 100644  
--- a/restart.c  
+++ b/restart.c  
@@ -125,6 +125,7 @@ struct ckpt_ctx {  
 } whoami;  
  
 int error;  
+ int success;  
  
 pid_t root_pid;  
 int pipe_in;  
@@ -677,9 +678,21 @@ int cr_restart(struct cr_restart_args *args)  
 if (global_feeder_pid)  
 waitpid(global_feeder_pid, NULL, 0);  
  
- if (ret < 0)  
+ if (ctx.error)  
     errno = ctx.error;  
  
+ if (ctx.success) {  
+ ckpt_dbg("c/r succeeded\n");  
+ ckpt_verbose("Restart succeeded\n");  
+ if (ctx.error)  
+     ckpt_verbose("Post restart error: %d\n", ctx.error);  
+ } else {  
+ ckpt_dbg("c/r failed ?\n");  
+ ckpt_perror("restart");
```

```

+ ckpt_verbose("Restart failed\n");
+ ret = -1;
+
+
return ret;
}

@@ -827,7 +840,7 @@ static int ckpt_pretend_reaper(struct ckpt_ctx *ctx)
    return ckpt_parse_status(global_child_status, 1, 0);
}

-static int ckpt_probe_child(pid_t pid, char *str)
+static int ckpt_probe_child(struct ckpt_ctx *ctx, pid_t pid, char *str)
{
    int status, ret;

@@ -840,17 +853,16 @@ static int ckpt_probe_child(pid_t pid, char *str)
    ret = waitpid(pid, &status, WNOHANG);
    if (ret == pid) {
        report_exit_status(status, str, 0);
-    errno = ECHILD;
-    return -1;
+    return ctx_ret_errno(ctx, ECHILD);
    } else if (ret < 0 && errno == ECHILD) {
        ckpt_err("WEIRD: %s exited without trace (%s)\n",
            str, strerror(errno));
-    return -1;
+    return ctx_set_errno(ctx);
    } else if (ret != 0) {
        ckpt_err("waitpid for %s (%s)", str, strerror(errno));
        if (ret > 0)
            errno = ECHILD;
-    return -1;
+    return ctx_set_errno(ctx);
    }
    return 0;
}
@@ -900,30 +912,39 @@ static int __ckpt_coordinator(void *arg)
    if (!ctx->args->wait)
        close(ctx->pipe_coord[0]);

-    return ckpt_coordinator(ctx);
+ /* set the exit status properly */
+    return ckpt_coordinator(ctx) >= 0 ? 0 : 1;
}

static int ckpt_coordinator_status(struct ckpt_ctx *ctx)
{

```

```

- int status = -1;
+ int status;
int ret;

close(ctx->pipe_coord[1]);
ctx->pipe_coord[1] = -1; /* mark unused */

ret = read(ctx->pipe_coord[0], &status, sizeof(status));
- if (ret < 0)
- ckpt_perror("read coordinator status");
- else if (ret == 0) {
- /* coordinator failed to report */
- ckpt_dbg("Coordinator failed to report status.");
- } else
- status = 0;

close(ctx->pipe_coord[0]);
ctx->pipe_coord[0] = -1; /* mark unused */

- return status;
+ if (ret < 0) {
+ ckpt_perror("read coordinator status");
+ return ctx_set_errno(ctx);
+ } else if (ret != sizeof(status)) {
+ /* coordinator failed to report */
+ ckpt_dbg("Coordinator failed to report status\n");
+ return ctx_ret_errno(ctx, EIO);
+ } else if (status != 0) {
+ /* coordinator reported failure */
+ ckpt_dbg("Coordinator reported error\n");
+ return ctx_ret_errno(ctx, status);
+ }
+
+ /* success ! */
+ ctx->success = 1;
+ return 0;
}

static int ckpt_coordinator_pidns(struct ckpt_ctx *ctx)
@@ -977,8 +998,8 @@ static int ckpt_coordinator_pidns(struct ckpt_ctx *ctx)
 * The child (coordinator) may have already exited before the
 * signal handler was plugged; verify that it's still there.
 */
- if (ckpt_probe_child(coord_pid, "coordinator") < 0)
- return ctx_set_errno(ctx);
+ if (ckpt_probe_child(ctx, coord_pid, "coordinator") < 0)
+ return -1;

```

```

ctx->args->copy_status = copy;

@@ -1017,8 +1038,8 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)
 * The child (root_task) may have already exited before the
 * signal handler was plugged; verify that it's still there.
 */
- if (ckpt_probe_child(root_pid, "root task") < 0)
- return ctx_set_errno(ctx);
+ if (ckpt_probe_child(ctx, root_pid, "root task") < 0)
+ return -1;

if (ctx->args->keep_frozen)
flags |= RESTART_FROZEN;
@@ -1028,20 +1049,15 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)
ret = restart(root_pid, ctx->args->infd,
flags, ctx->args->klogfd);

- if (ret < 0) {
- ckpt_perror("restart failed");
- ckpt_verbose("Failed\n");
- ckpt_dbg("restart failed ?\n");
- return ret;
+ if (ret >= 0) {
+ ctx->success = 1; /* restart succeeded ! */
+ ret = 0;
}

- ckpt_verbose("Success\n");
- ckpt_dbg("restart succeeded\n");
-
- ret = 0;
-
if (ctx->args->pidns && ctx->tasks_arr[0].pid != 1) {
/* Report success/failure to the parent */
+ if (ret < 0)
+ ret = ctx->error;
if (write(ctx->pipe_coord[1], &ret, sizeof(ret)) < 0) {
ckpt_perror("failed to report status");
return ctx_set_errno(ctx);
@@ -1068,11 +1084,6 @@ static int ckpt_coordinator(struct ckpt_ctx *ctx)
ret = ckpt_collect_child(ctx);
}

- if (ret < 0)
- ckpt_dbg("c/r failed ?\n");
- else
- ckpt_dbg("c/r succeeded\n");
-
```

```
    return ret;  
}
```

--  
1.7.1

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---