TEDA - DEPLOYMENT SCRIPTS FOR ERLANG APPLICATIONS

This document introduces the deployment scripts that are used to deploy Erlang applications to remote machines located at the CDC-Lab of the University of Fribourg. The whole environment is referred to as teDA for "test and evaluation for distributed algorithms".

1. SETUP

1.1 Scripts' Philosophy

Several POSIX compatible shell scripts are provided with the teDA environment. They are written in a modular way, i.e., some scripts rely on calling other scripts for basic manipulations. The scripts are kept as short as possible and are well documented. Each script contains a small documentation at the top, including:

- a description of what the script does,
- the usage information of the script (i.e. required / optional parameters),
- some example uses.

The name of the script gives an idea of what kind of manipulation it does. Each script is specialized for doing a certain manipulation, e.g. `run.sh` runs an Erlang application from a given remote master node.

We encourage you to read and understand these shell scripts in order to be able to adapt them to your needs if necessary.

Important notes about example commands

For all example commands in this document, we take the hypothesis that the current working directory is an application folder (e.g. `teda/apps/echo)` and that the command is launched from this folder.

Example commands are put inside a gray-shaded bordered box similar to this one.

1.2 Directory Structure

Local directory structure

The Erlang environment on your local machine is structured as follows:

```
<your_local_base_directory>
   |- apps/{echo/, sieve/, cm/, ...}
   |- conf/{hosts.conf, teda.conf, ...}
   |- doc/{cdc_lab_memento.pdf, scripts_memento.pdf,...}
   |- scripts/{...}
```

The `apps/` folder contains Erlang applications. Each application should be in its own folder. Each application folder, e.g. `echo/`, is typically structured as:

```
echo/{*.erl, *.io, *.beam, *.pdf} |
     {src/*.erl, io/*.io, bin/*.beam, doc/*, Makefile},
```

the second structure being necessary if the documentation is created by Erlang's documentation generator *EDoc.*

The `conf/` folder contains configuration files used by the scripts:

- `teda.conf`: global variables used by all scripts. In particular, contains your unifr username and reference to your RSA key.
- `hosts.conf`: list of hosts of the CDC-Lab that are in the UNIFR domain 134.21.72.* and can be accessed from within the university's network (i.e., secure-unifr or VPN).

The `doc/` folder contains documentation of the teDA environment. Note: the documentation about an application should be located in its own application folder, e.g. `apps/echo/`.

The `scripts/` folder contains the following scripts:

Configuration scripts

- `RSA_setup.sh`: generates and / or distributes RSA key pairs on hosts
- `RSA_destroy.sh`: deletes RSA keys locally and on hosts

Main scripts

- `ping.sh`: pings a set of remote hosts and returns a list of “living” hosts
- `depl_app.sh`: deploys an app on a set of hosts machines
- `depl_enodes.sh`: starts Erlang nodes on a set of hosts machines and returns their ids and the cookie associated with them
- `run.sh`: runs an app on a set of hosts machines
- `run_local.sh`: runs an app locally on your machine (localhost)

Miscellaneous scripts

- `emergency.sh`: kills all Erlang processes on a set of hosts machines

Helper scripts

- all the other scripts are helper scripts that are required by the scripts above. Comments in the scripts explain what they do. They will not be detailed any further here.

Remote directory structure

When you deploy applications to remote hosts, the following directory structure is created on remote hosts:

```
~
|- mpe
    |- erl
       |- teda
            |- apps {echo/, sieve/, cm/, ...}
```

`mpe` stands for "my programming environment", `erl` for Erlang, `teda` for "test and evaluation for distributed algorithms". Each application is deployed in its own folder inside `apps/`.

1.3 Configuration

Before using the bash shell scripts for the first time, you have to take the following configuration steps:

Scripts’ privilege

Make sure that the executable bit is set for all scripts in `scripts/*.sh`. To set it, type:

```
chmod +x ../../scripts/*.sh
```

Configure teda.conf

In file `conf/teda.conf`, update the line `UNIFR_USER=user` by changing `user` to your unifr username (e.g. `evequozf`).

Configure hosts.conf

In the file `conf/hosts.conf`, change all instances of the name `user` to your unifr username.

Deploy RSA keys

In order to avoid typing your password multiple times when deploying and running an application, you can generate an RSA key and distribute it to the host machines.

```
../../scripts/RSA_setup.sh  hosts.conf
```

If needed, you can delete previously distributed keys by running `RSA_destroy.sh` (see documentation inside that file).

2. DEPLOY AND RUN ERLANG APPLICATION ON REMOTE HOSTS

Deploying and running an Erlang application on 1 or 2 hosts with 1 or 2 enodes each can be done manually (cf. [BH 15] and [Arm 13, Chap. 14]). For more hosts or more enodes, this manual way is too cumbersome. It is better so use some scripts. In the following we propose a couple of shell scripts.

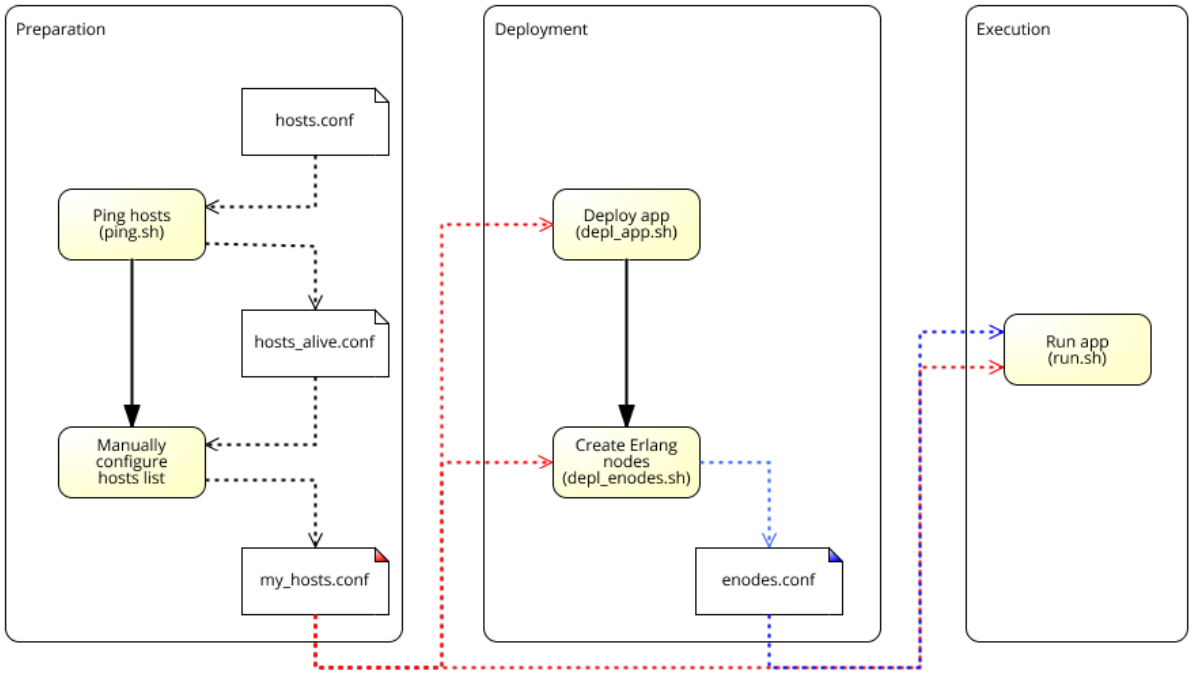

Fig. 1: Procedure to deploy and run an Erlang application on remote hosts.

The overall procedure for preparing, deploying and running an Erlang application on remote hosts is pictured in Fig. 1. In that figure, yellow boxes are the actions that you have to take and *.conf items are files needed or produced by the scripts. The three steps of the procedure are detailed below.

2.1 Preparation

Ping hosts (ping.sh)

First, you need to ping remote hosts to check which hosts are up and running. The script `ping.sh` takes an app and a `conf/hosts.conf` file given as argument and checks if the hosts listed there are running correctly. This command creates a file named `hosts_alive.conf` that contains only 'live' hosts. This file is created in the application folder of the application given in parameter.

For example, here is a command that reads the list of hosts from `conf/hosts.conf` and writes the list of 'live' hosts in `apps/echo/hosts_alive.conf`:

```
../../scripts/ping.sh echo hosts.conf
```

Manually configure hosts list

You can then modify the file `hosts_alive.conf` by hand to retain only the machines that you want to deploy to. You can also set the number of Erlang nodes that must be started on each machine. Consider for example the following content for a `hosts_alive.conf` file:

```
# Sample manually modified hosts_alive.conf
diufmac33.unifr.ch evequozf 1 262144
diufmac31.unifr.ch evequozf 2 262144
```

Each line contains, from left to right:

- the IP/DNS name of the remote host
- the username to connect on the remote host
- the number of Erlang nodes to start on that host
- the maximum number of Erlang processes allowed on that host (no need to change)

The lines in the example above mean that we want to deploy the application on two machines (`diufmac33` and `diufmac31`) with user `evequozf`. Then, `1` Erlang node should be started on `diufmac33` and `2` nodes on `diufmac31`. A maximum number of `262144` Erlang processes is allowed on each machine.

2.2 Deployment

In order to be able to deploy your application on remote hosts, you first need to have configured a list of 'live' hosts (`hosts_alive.conf`) as described above. The script `depl_app.sh` takes as parameter an application, reads a list of hosts (by default, it reads the file `hosts_alive.conf` in the application folder) and copies the application folder on all hosts listed. The script can be used to simply copy the application folder, but also to build it on remote hosts (with option `make`) or remove it altogether from remote hosts (with option `clean`).

For example, here is the command to deploy the content of the application in `apps/echo/` to the hosts listed in default file `apps/echo/hosts_alive.conf` and build it on remote hosts:

```
../../scripts/depl_app.sh  echo  make
```

This script accepts alternative parameters (e.g. providing a custom list of hosts different from the default `hosts_alive.conf`). Please read the documentation in the script itself for further information.

2.3 Execution

Once the application is deployed on remote hosts you can execute it. This happens in two sequential steps.

Create Erlang nodes (enodes.sh)

First, you need to start Erlang nodes on the remote hosts. As previously you need the list of hosts that you have deployed your application to (`hosts_alive.conf`). This file also contains the number of Erlang nodes that must be started on each remote host. To start the Erlang nodes, use the script `depl_enodes.sh` and provide as arguments the name of the application folder and the name of the hosts list to use. After execution, this script creates a file `enodes.conf` in the application folder that contains the 'ids' of the created Erlang nodes and the 'cookie' that will allow their common execution.

For example, here is the command to start the Erlang nodes for the application in `apps/echo/` according to the hosts list in `apps/echo/hosts_alive.conf`:

```
../../scripts/depl_enodes.sh  echo  hosts_alive.conf
```

This creates a file `enodes.conf` in the application folder that looks like this:

```
603363030.
'690663826@diufmac33.unifr.ch'.
'58107335@diufmac31.unifr.ch'.
'2179084361@diufmac31.unifr.ch'.
```

The first line of this file contains the 'cookie'. The following lines contain the 'ids' of the Erlang nodes created.

Run application (run.sh)

Once the Erlang nodes are created, you can launch your application. This can be done with the `run.sh` script, which takes as argument the name of the application to run, the Erlang command to send to the master node, the machine to use as master, and the username to connect to that machine. This file reads by default the hosts list from `hosts_alive.conf` and the cookie and ids of Erlang nodes from `enodes.conf` in the application folder. See the documentation inside the script for detailed usage options.

For example, here is the command to run the application `echo` by calling the Erlang function `echod_nn:master()` and according to the hosts list in `apps/echo/hosts_alive.conf`, using `diufmac33.unifr.ch` as master (with user `evequozf`):

```
../../scripts/run.sh echo \
                     "echod_nn:master()" \
                     hosts_alive.conf \
                     diufmac33.unifr.ch \
                     evequozf
```

3. MISCELLANEOUS

3.1 Compact example: deploy and run the 'echo' application

Sequence of commands and actions to prepare, deploy and run the echo application with default options, and remove it from remote hosts after running (example):

```
   --- prepare a hosts config file, e.g. conf/my_hosts.conf
$ ../../scripts/ping.sh echo my_hosts.conf
   --- update produced "alive hosts" if necessary
   --- i.e. apps/echo/hosts_alive.conf
$ ../../scripts/depl_app.sh echo make my_hosts.conf
$ ../../scripts/depl_enodes.sh echo hosts_alive.conf enodes.conf
$ ../../scripts/run.sh echo \
                       "echod_nn:master()" \
                       hosts_alive.conf \
                       diufmac33.unifr.ch \
                       evequozf
$ ../../scripts/depl_app.sh echo clean
```

3.2 Useful commands

The following commands can be used when connected on remote hosts to check what happens there.

- Return the ids of running Erlang nodes for the current user on the current machine:

```
$ epmd -names
```

- Return the processes running for user `evequozf` (note: Erlang nodes run as separate `beam.smp` processes):

```
$ top -u evequozf
```

- Kill all running Erlang nodes on the current machine for the current user

```
$ killall -9 beam.smp
```

References

[BH 15] Béat Hirsbrunner: “CDC-Lab_Memento.pdf”, University of Fribourg, September 2015.
[Arm 13] Joe Armstrong: “Programming Erlang”, 2nd ed., The Pragmatic Bookshelf, 2013.