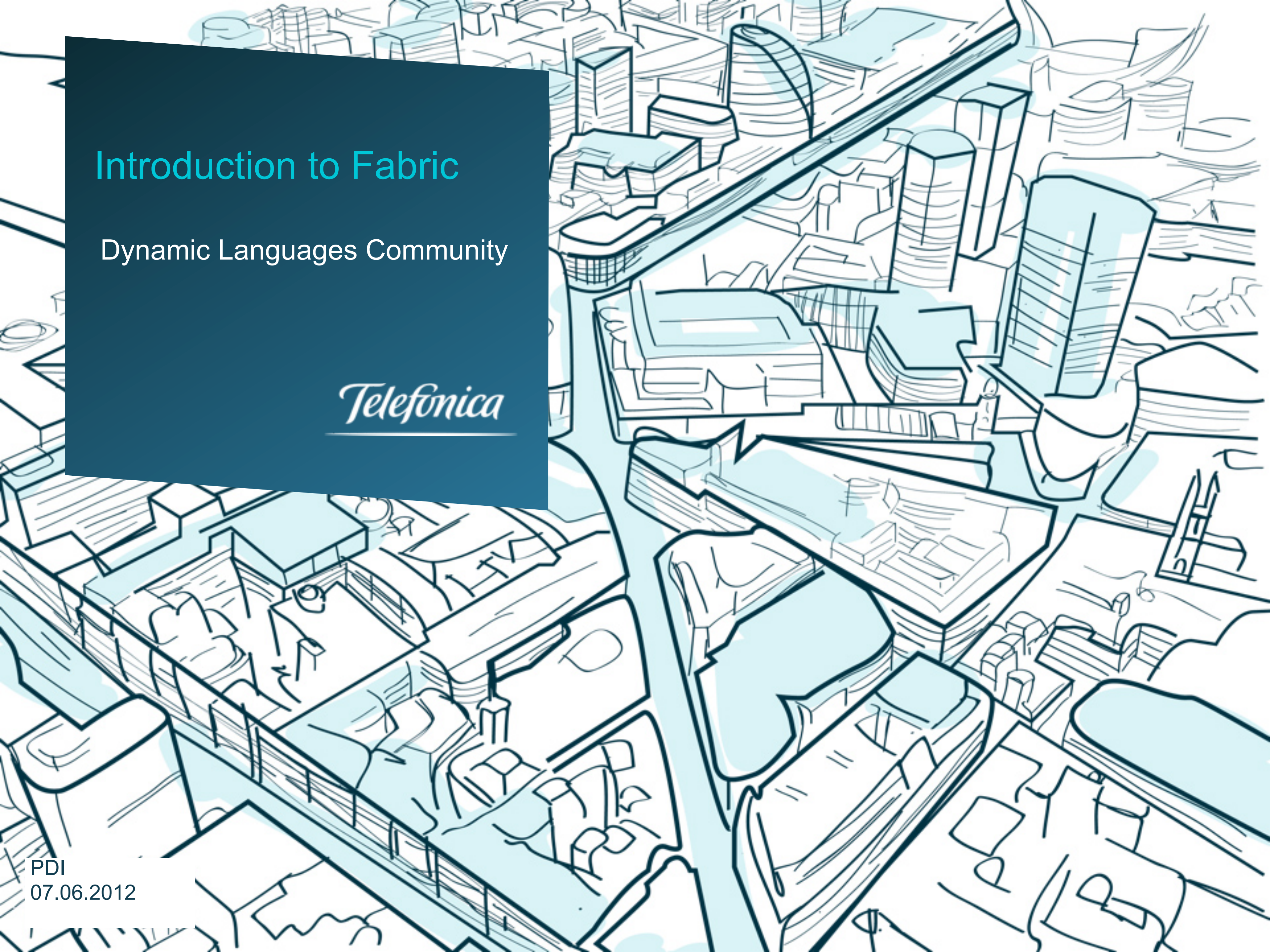

Introduction to Fabric

Dynamic Languages Community

Telefónica

PDI
07.06.2012

What is Fabric?

- What fabfile.org says:
  - "Fabric is a Python (2.5 or higher) library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks."
  - "It provides a **BASIC** suite of operations for executing local or remote shell commands (normally or via sudo) and uploading/downloading files, as well as auxiliary functionality such as prompting the running user for input, or aborting execution."
- What I say:
  - It is a bunch of commands, that's all
    - To define scripts with tasks
    - With good support to operate on several remote hosts

What it is not?

- A packaging tool
- A deployment tool
- A SysAdmin tool
- A set of rich commands such as “pack as ZIP, deploy code, deploy DB tables and restart OS services”

- **But your fabfile could be all of these tools and much more…**

Simplest task

```python
from fabric.api import run

def host_type():
    '''Get host type'''
    run('uname -s')
```

```
$ pip install fabric
...
$ fab host_type -H iammy01,int-iammy -u iammy
[iammy01] Executing task 'host_type'
[iammy01] run: uname -s
[iammy01] Login password for 'iammy':
[iammy01] out: Linux

[int-iammy] Executing task 'host_type'
[int-iammy] run: uname -s
[int-iammy] out: Linux


Done.
Disconnecting from int-iammy... done.
Disconnecting from iammy01... done.
```

Useful fab command line arguments

- **-h** / **--help**: help :D
- **-f** / **--fabfile**: fabfile to be used
- **-u** / **--user**: username to use to log into remote hosts
- **-H** / **--hosts**: comma-separated list of hosts to operate on
  - Alternatively, comma-separeted list of **user@host**
- **-l** / **--list**: list current fabfile tasks
- **-R** / **--roles**: comma-separated list of roles to operate on
- **--abort-on-prompts**: useful when automating

Main commands

- **run**: execute a command in remote hosts
  - **sudo**: idem but using sudo to get superuser privileges (signature differs a bit)
  - **local**: idem but in localhost (signature differs a bit)
- **execute**: run a task from another task
- **put / get**: send / retrieve files to / from remote host
- **cd** / **lcd**: change current remote / local working directory
- **prompt**: interact with the user through command line
- **abort**: halt Fabric execution
- **require**, **reboot**, **open_shell**…

Environment variables / settings

- **env**: “a Python dictionary subclass which is used as a combination settings registry and shared inter-task data namespace”
  - **“environment” variables dictionary**
  - **singleton**
  - **thread-safe (?)**
  - **you can add your custome settings variables**

- **settings**: context manager to temporarily change environment variables

Hosts and roles lists

- **env.hosts**: environment variable with hosts list, built with:
  - **Direct declaration (3)**

    ```
    env.hosts = ['int-iammy', 'iammy01']
    ```

  - **-H** / **--hosts** command line arguments **(4)**
  - Per task via command line **(1)**

    ```
    $ fab listdir:folder=/tmp,hosts=root@iammy01
    [root@iammy01] Executing task 'listdir'
    Listing folder...
    [root@iammy01] run: ls /tmp
    ...
    ```

  - Per task via decorator **(2)**
- **env.roledefs**: roles of hosts dictionary

  ```
  env.roledefs = {'myservers': ['int-iammy', 'iammy01'],
                  'myhost': ['localhost']
                 }
  ```

  - There is also a **roles** decorator

Tasks decorators

- **@hosts**: to specify in which host must be executed a task. Useful for tasks to be run once in local:
  - `@hosts('localhost')`
- **@task**: to specify which function is a Fabric task. The rest of functions can not be lauched!
- **@parallel / @serial**: to specify how a tasks may be run:
  - `@parallel(pool_size=5)`
- **@with_settings**: change environment variables only in the task
  - `@with_settings(warn_only=True)`

Tasks command line parameters

```python
from fabric.api import run

@task
def listdir(folder='.'):
    '''List given folder content'''
    print 'Listing folder...'
    run('ls {0}'.format(folder))
```

```
$ fab listdir
[int-iammy] Executing task 'listdir'
Listing folder...
[int-iammy] run: ls .
...

$ fab listdir:/tmp
[int-iammy] Executing task 'listdir'
Listing folder...
[int-iammy] run: ls /tmp
...

$ fab listdir:folder=/tmp
[int-iammy] Executing task 'listdir'
Listing folder...
[int-iammy] run: ls /tmp
```

Telefónica