Juju Charm Development

EAPCONNECT WORKSHOP 2019

Claudio Pisa, Matteo Di Fazio, Marco Lorini, Delia Passalacqua

GARR - Distributed Computing and Storage Department - claudio.pisa@garr.it

Charm concepts

Charm

- deployment "brick"
- a collection of scripts with some metadata
- can be written in any scripting/programming language
- can expose configuration parameters
- Juju orchestrates the deployment, composition and scaling of Charms
- a deployed Charm is called an **application**
- an instance of an application installed on a specific machine is called a **unit**

Charm concepts

Events

- Occur during lifetime of services
- Events: install, configure, start, upgrade, stop...

Hook

- Steps to be performed to handle an event
- hooks are executable files in a charm’s hooks directory
    - can be written in any scripting/programming language
- hooks with particular names will be invoked for specific events

Charm concepts

Charm relations

- establish a connection between services
- configuration parameters flow between charms through relation interfaces

Charm interfaces

- define how services interact with each other
- one charm is the provider, like a socket
- any charm can consume the interface, like a plug

Hooks

- **install** - perform one-time setup operations
- **config-changed** - run on configuration changes
- **start** - used to ensure the charm’s software is running
- **upgrade-charm** - runs after an upgrade operation
- **stop** - runs before the end of the unit destruction
- **update-status** - run by Juju at regular interval

Charm lifecycle events vs hooks

- ***Event*** *(command) --> hooks*

- **deployment** (juju deploy) --> install, config-changed, start
- **configuration** (juju config)--> config-changed
- **upgrade** (juju upgrade-charm) --> upgrade-charm, config-changed
- **destroy** (juju remove-{application, unit}) --> stop
- **periodically** --> update-status

Charm lifecycle events vs hooks

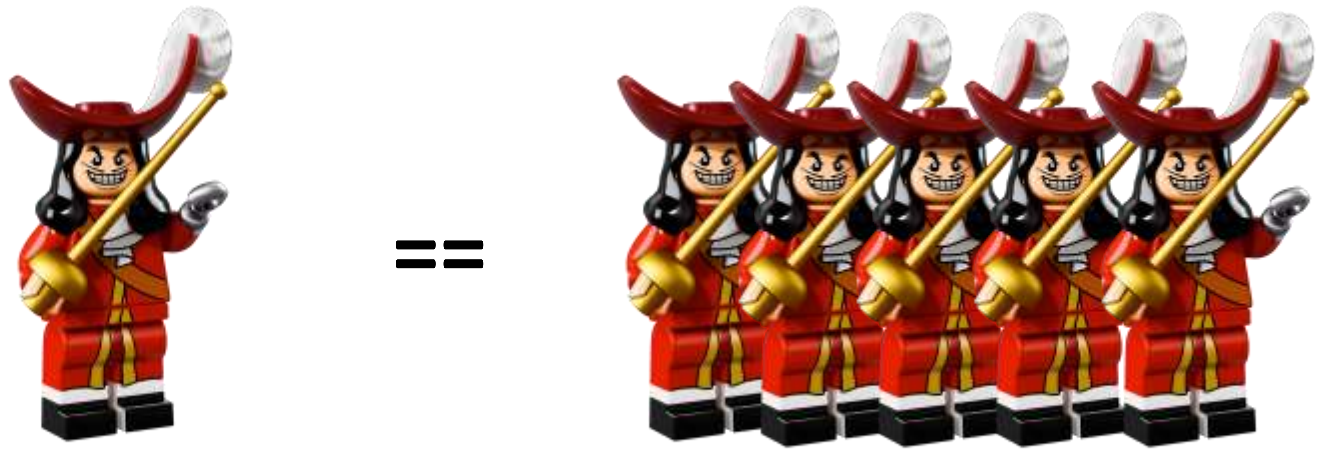
- **joining a relation** (juju add-relation) --> x-relation-joined, x-relation-changed
- **leaving a relation** (juju remove-relation) --> x-relation-departed, x-relation-broken

https://jaas.ai/docs/charm-hooks

Consortium GARR THE ITALIAN EDUCATION & RESEARCH NETWORK

Hooks

- All the hooks must be written to be **idempotent**
  - there should be no difference from running the hook once from running it multiple times



Hook invocation

- Whenever a hook-worthy event takes place:
  - the unit tries to find a **hook with the right name**
  - if the hook doesn’t exist, the agent continues without complaint
  - if the hook does exist, it is invoked
    - without arguments
    - in a specific hook context
    - its output is written to the unit's log
    - if it returns a non-zero exit code, the agent enters an error state and awaits user intervention

Charm tools

Pre-defined functions can be used in hooks:

- **status-set** - status can be: maintenance, blocked, waiting, active
- **juju-log** - write into the logs
- **config-get** - returns a key-value mapping of the current configuration settings
- **open-port** - marks a TCP or UDP port as appropriate to open
- **relation-set** - communicate configuration parameters to related applications.
  - by convention the charm that provides an interface is likely to set values, and a charm that requires that interface will read them
- **relation-get** - reads the configuration parameters set by the other charm in the relation

https://jaas.ai/docs/charm-writing/hook-env

Charm's anatomy

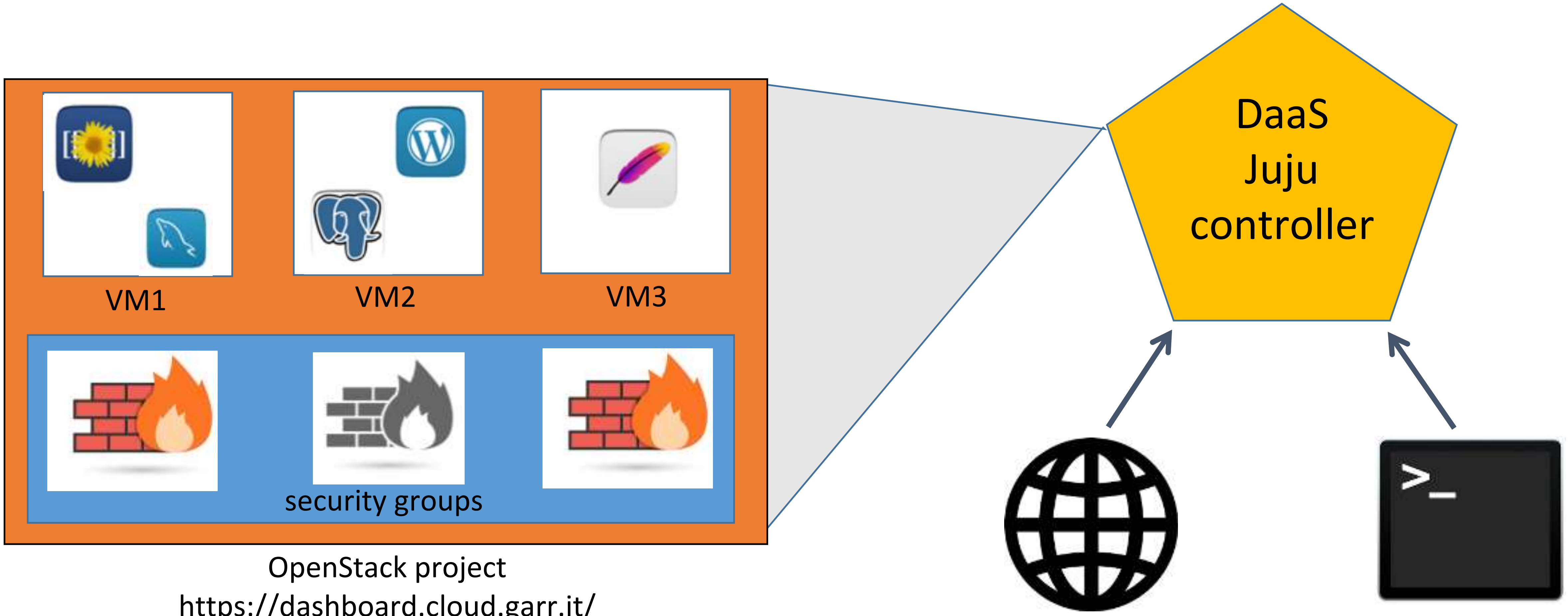
```
simplecharm/
├── README.md
├── config.yaml
├── hooks
│   ├── config-changed
│   ├── install
│   ├── relation-name-relation-broken
│   ├── relation-name-relation-changed
│   ├── relation-name-relation-departed
│   ├── relation-name-relation-joined
│   ├── start
│   ├── stop
│   └── upgrade-charm
├── icon.svg
├── metadata.yaml
└── revision
```

- README.md: basic documentation
- **config.yaml**: charm configuration parameters
- **hooks/** : hook executables directory
- icon.svg : an icon for the application
- **metadata.yaml**: Charm metadata
- revision: Charm versioning information

hands on

Charm Development

DaaS (Deployment as a Service)



OpenStack project
https://dashboard.cloud.garr.it/

GUI
https://daas-playground.cloud.garr.it

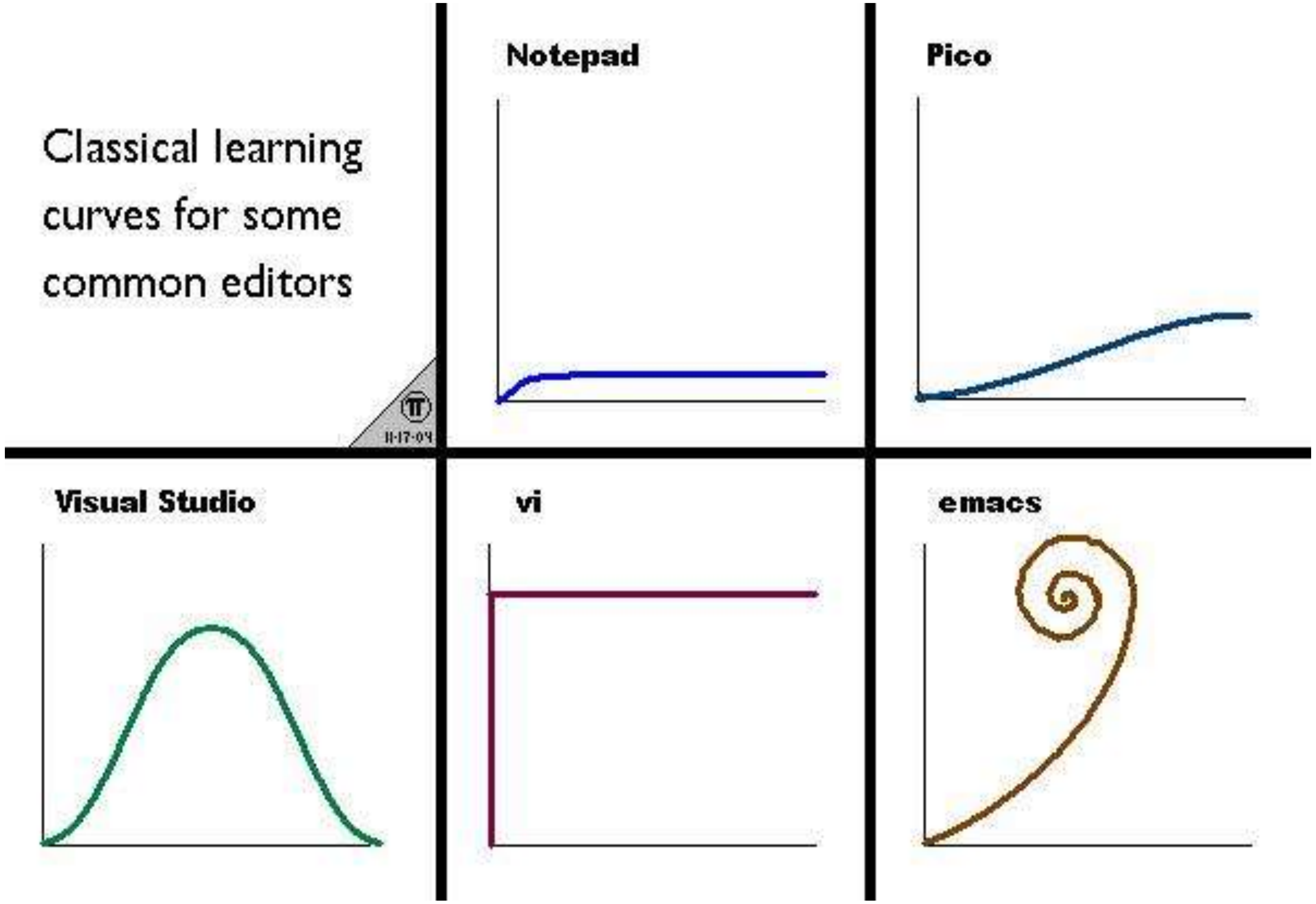
CLI
```
ssh playerXXX@playground-cli.cloud.garr.i
```

Wordpress

- WordPress (WordPress.org) is a content management system (CMS)
- based on PHP and MySQL
- plugin architecture
- template system
- it is most associated with blogging (but supports other types of web content)
- used by more than 60 million websites
- the most popular website management system in use

https://en.wikipedia.org/wiki/WordPress

Classical learning curves for some common editors



https://garr.workplace.garr.it/sh/X5r

hands on

Reactive Charms

Reactive Framework

- managing distributed software is difficult
- the exact action to take in response to an event can depend on which events have happened in the past
- charms.reactive represents a system for setting flags with semantic meaning and then driving behavior off of the combination of those flags

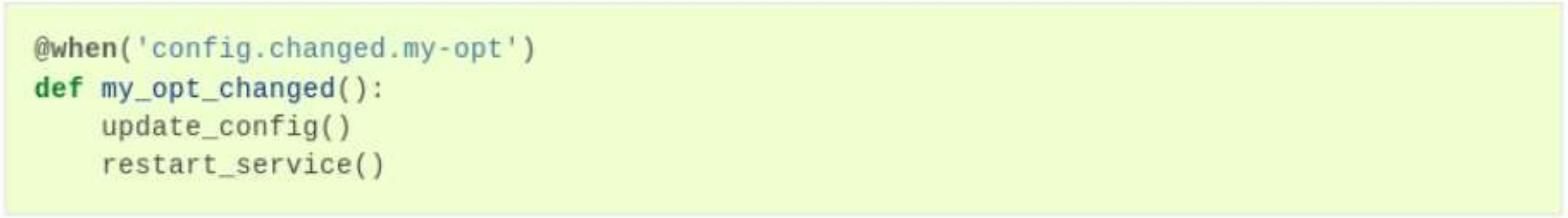
Reactive Framework

- use decorators to indicate that blocks of code "react" to certain combination of conditions, corresponding to a logical combination of flags
- example decorators:

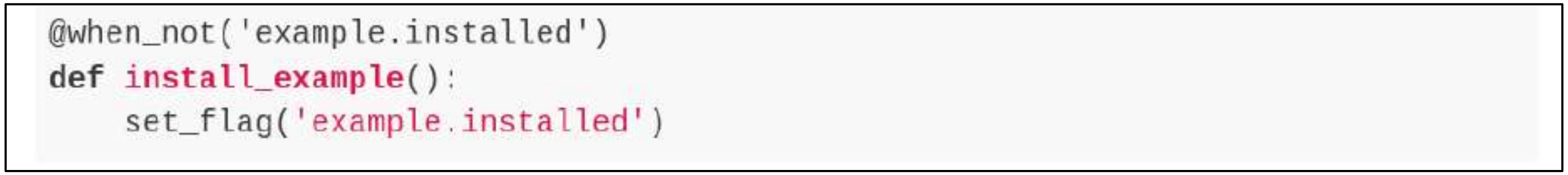
```
@when()

@when_any()

@when_all()

@when_not()

@when_file_changed()

@hook()
```

Setting Flags

- Flags are string identifiers linked to a condition
  - e.g. “config.changed.myopt”, “myapp.configured”
- Flags can be set either:
  - programmatically: through the set_flag() function
  - automatically by Juju
    - https://charmsreactive.readthedocs.io/en/latest/managed-flags.html
  - e.g. ‘config.changed.myopt’



```python
@when('config.changed.my-opt')
def my_opt_changed():
    update_config()
    restart_service()
```



```python
@when_not('example.installed')
def install_example():
    set_flag('example.installed')
```

```python
from charms.reactive import set_flag, clear_flag, when
from charms.reactive.helpers import any_file_changed
from charmhelpers.core import templating, hookenv

@when('db.database.available', 'config.set.admin-pass')
def render_config(pgsql):
    templating.render('app-config.j2', '/etc/app.conf', {
        'db_conn': pgsql.connection_string(),
        'admin_pass': hookenv.config('admin-pass'),
    })
    if any_file_changed(['/etc/app.conf']):
        set_flag('myapp.restart')

@when('myapp.restart')
def restart_service():
    hookenv.service_restart('myapp')
    clear_flag('myapp.restart')
```

Reactive Charm Tutorial

https://cloud.garr.it/charms/create-and-deploy-charms/index.html

Thank You

Claudio Pisa - claudio.pisa@garr.it