UD04A03.–
Eines per al desplegament automàtic d'aplicacions.
Deployer.

Índex

1. Introducció.

El desplegament d'una aplicació web és una tasca complexa i variada depenent de la configuració i dels requisits de cada aplicació. Com hem vist al llarg del curs, les tasques típiques inclouen:

- **Desplegar el codi** en el servidor de producció, **staging**, **desenvolupament** (via SSH, FTP, HTTPS, GIT…).
- **Instal·lar** les dependències externes i internes de l'aplicació (Composer, Phar…).
- Dur a terme les **migracions** de la **base de dades** amb les possibles modificacions que s'hagen dut a terme en el model de dades.
- Neteja i creació de cau (rutes, classmap…).

A més, també haurem de tindre en compte altres tasques addicionals com:

- Comprovar si el servidor compleix amb els **requeriments necessaris** per al desplegament de la nostra aplicació
- Creació d'una **àrea de preparació** on construir el codi abans de la seua publicació. (mimificació, execució de tests unitaris, compilació, construcció...)
- Manteniment de diferents versions en el servidor que ens permetran la marxa arrere.
- **Gestionar els fitxers** que han de mantenir-se entre desplegaments (configuració d'entorns (**.*env**), sessions, documents i imatges pujades pels usuaris...)
- Configuració de les **variables d'entorn** de l'aplicació: **credencials** d'accés a BBDD, nivell de log, entorn, comptes per a l'enviament de mail…
- Tasques específiques que es duran a terme a cada entorn de desplegament (staging, development, test…).

> L'execució de cadascuna d'aquestes tasques de manera manual comporta una **pèrdua de temps** considerable, alhora que augmenta la probabilitat de cometre errors a causa de la introducció de factor **humà** en el procés de desplegament.

Totes aquestes tasques podríem automatitzar-les mitjançant la construcció d'un **script Bash** (o similar), no obstant això, existeixen una sèrie d'eines ben provades amb tasques predefinides que ens ajudaran a la seua **automatització**.

En aquesta pràctica aprendrem a desplegar una aplicació **Laravel** de manera automàtica amb **zero downtime** i mantenint **còpies de seguretat** de les versions desplegades a través de l'eina **Deployer.**

1.1 Prerequisits.

Assumim que l'alumne ja disposa d'un **servidor Ubuntu Server 22.04** amb un entorn **LEMP** instal·lat i configurat al llarg de les pràctiques anteriors. A més haurem de tindre instal·lat en el **servidor de desplegament:**

- **Servei SSH** amb accés des de la màquina local.

- **Client Git** al terminal**.**

2. Eines d'automatització.

Encara que ens centrarem en el desplegament de l'aplicació mitjançant el framework Deployer, haurem de tindre en compte que existeix una gran quantitat d'eines afins i que l'elecció d'una o altra dependrà entre altres de:

- Els coneixements dels desenvolupadors.
- La tecnologia utilitzada per a la implementació de l'aplicació (PHP, Java, Ruby).
- Les necessitats de cada aplicació.

A continuació enumerarem alguna de les eines més característiques:

- Deployer: Aplicació nativa de Php per al desplegament d'aplicacions amb tasques preconfigurades per als principals frameworks.
- Capistrano: Eina d'automatització i desplegament desenvolupat en **Ruby**.
- Magallanes: Eina d'automatització desenvolupada en **Php**.
- Maven: Gestor de dependències i eina per a l'automatització del desplegament i construcció d'aplicacions. Està desenvolupat en J**ava**.

Activitat 1

Busca informació en les pàgines oficials de cadascuna de les eines exposades i indica les tecnologies i frameworks més característics amb els quals s'associa per a desplegament d'aplicacions.

3. Deployer. Preparació de l'Entorn

3.1 Creació de l'usuari i directori de desplegament en el servidor

Abans d'utilitzar Deployer per a desplegar el projecte, **deurem configurar el servidor** sobre el qual es durà a terme el desplegament. Definirem un nou usuari, amb els permisos mínims necessaris per al desplegament de l'aplicació, de manera que **Deployer**

el farà servir **per a connectar-se des del client** i executar les tasques necessàries per al desplegament de l'aplicació. Els passos a seguir són els següents:

1. Generar el nou usuari. Com a exemple utilitzarem **ddaw-ud4-deployer.**

```
$ sudo adduser ddaw-ud4-deployer
```

2. Necessitem que tots els **fitxers generats** durant el **desplegament** tinguen permisos de lectura **i escriptura** per a l'usuari ddaw-ud4-deployer i que **nginx** i el servidor **php-fpm puguen llegir** en ell, per la qual cosa establirem un **umask** de 022 per a l'usuari generat.

```
$ sudo chfn -o umask=022 ddaw-ud4-deployer
```

3. Per a finalitzar generarem el directori arrel on es desplegarà l'aplicació i canviarem el seu propietari.

```
$ sudo mkdir /var/www/ddaw-ud4-a4/html -p
$ sudo chown ddaw-ud4-deployer:www-data /var/www/ddaw-ud4-a4/html
```

4. Necessitem que tots els directoris i subdirectoris que es creen durant el desplegament siguen del grup **www-data.** D'aquesta manera podrem assignar al servei **php-fpm** permisos específics d'escriptura sobre determinats directoris, i per això hem d'activar el bit **group-id** del directori arrel perquè els subdirectoris que es creuen nous hereten el grup del directori pare.

```
$ sudo chmod g+s /var/www/ddaw-ud4-a4/html
```

5. Per finalitzar, haurem d'instal·lar l'eina **acl** perquè Deployer puga gestionar permisos més específics en alguns directoris del projecte.

```
$ sudo apt install acl
```

Activitat 2

Amb quins permisos, propietari i grup es generaran els fitxers i directoris quan s'utilitze l'usuari ddaw-ud4-deployer una vegada aplicada la màscara?

3.2 Instal·lació de l'entorn en la màquina client

Una vegada tenim configurats els directoris i creats l'usuari de desplegament en el servidor, **començarem a configurar el client** des del qual **s'iniciarà el procés de desplegament.** Per fer-ho, obrirem el nostre Ide amb el projecte que tinguem desplegar i, **fent ús de la ferramenta composer**, afegirem la dependència de deployer a la nostra aplicació.

```
composer require --dev deployer/deployer
```

Pots obtindre més informació en la seua pàgina oficial. A partir d'aquest moment tindrem l'eina **dep** disponible al sistema.

```
profe@client-desenvolupament$ vendor/bin/dep
Deployer 7.3.3
Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
  --ansi
  --no-ansi             Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
  -f, --file[=FILE]
  -v|vv|vvv, --verbose Increase the verbosity of messages:
                        1 normal output, 2 for habite verbose output and 3 for debug
Available commands:
  autocomplete    Install command line autocompletion capabilities
  build
  cleanup         Cleaning up old releases
  deploy          Deploy your project
  help            Displays help for a command
  init            Initialize deployer in your project
  list            Lists commands
  rollback        Rollback to previous release
...
```

Si estàs desenvolupant la teua aplicació mitjançant un sistema **operatiu Windows,** serà més recomanable la instal·lació de Deployer de manera local al projecte a través

de Composer. En la **documentació oficial** tens informació més detallada sobre com dur a terme el procés. En aquest cas hauràs de consultar els canvis en el fitxer de deploy.php.

3.3 Configuració de Deployer al projecte Laravel.

Una vegada configurada les diferents eines necessàries perquè **Deployer** puga funcionar, **configurarem Deployer perquè treballe al costat de Laravel** al desplegament de l'aplicació.

Hem de destacar que **tots els passos** descrits a continuació **s'executaran en la màquina client**, i serà **Deployer** qui s'encarregarà de dur **a terme** les **connexions** amb el **servidor** par dur a terme el seu desplegament.

Accedirem el nostre **IDE** de dessenvolupament, clonarem el projecte i obrirem una terminal. Seguidament, e**ns situarem en el directori arrel** del projecte Laravel i inicialitzarem **Deployer**.

```
$ vendor/bin/dep init
```

Contesta les preguntes que vagen sortin, D'aquesta forma se te generarà un fitxer `deploy.php` personalitzat. **Si alguna resposta no la coneixes** encara, **pots modificarla després** manualment editant el fitxer `deploy.php` generat.

A partir d'ara disposarem d'un fitxer deploy.php , que será el que done instruccions a deployer sobre com desplegar l'apliicació.

Al haver seleccionat laravel com a framework, ja disposem d'una **plantilla** bàsica que conté les configuracions mínimes per al **desplegament d'un projecte** desenvolupat amb el framework**.**

Aquesta plantilla té una sèrie de tasques configurades com són: otorgar permisos d'escriptura als directoris cau (cache) **i log** que Laravel necessita per a la seua execució, buidar la cache de manera prèvia a la publicació, instal·lació de les dependències…

D'aquesta manera, només ens preocuparem d'establir **el repositori on resideix el projecte** a desplegar, el **nom del host** o la **IP del servidor** de desplegament i **l'usuari del servidor** que volem utilitzar per a dur a terme les tasques de desplegament.

El nou fitxer deploy.php formarà part del nostre projecte i ha d'estar disponible per a qualsevol dels desenvolupadors que clone el projecte, pel que una vegada funcione correctament, farem un nou commit que l'incloga i el pujarem al repositori.

deploy.php

```
<?php
        namespace Deployer;
        require 'recipe/laravel.php';
        // Nom del Projecte
        set('application', 'NomDelProjecte');
        // Repositori on es troba el nostre projecte
        set('repository', 'https://github.com/acoloma-edu/todo');
        // [Opcional] Assignar tty for git clone (escriptura en eixida estàndard durant el clonat)
        set('git_tty', true);
        // Establim els fitxers i directoris que han de mantindre's durant els desplegaments per defecte
        //heretem de la plantilla el directoris bootstrap/cache','storage' i els fitxer .env
        add('shared_dirs', ['']);
        add('shared_files', ['']);
        // Establim els directoris amb permisos d'escriptura pel servidor d'aplicacions heretem de la plantilla
        // el directoris bootstrap/cache','storage''
        add('writable_dirs', []);
        // Establim el path on ha de ser desplegada la nostra aplicació
        host('your_server_ip') ->set('remote_user', 'ddaw-ud4-deployer'')
                               ->set('identity_file', '~/.ssh/id_rsa')
                               ->set('deploy_path', '/var/www/ddaw-ud4-a4/html');
        // Podem definir diferents tasques que s'executaran durant, abans o després de cada
        desplegament
        task('build', function () {
            run('cd {{release_path}} && build');
        });
        // Si el desplegament falla, automàticament fem un unlock i deixem la versió anterior.
        after('deploy:failed', 'deploy:unlock');
        // Executem la migració de la base de dades just abans de dur a terme l'enllaç simbòlic a la nova
        versió. La primera vegada que fem un desplegament no disposarem de l'arxiu .env, per la qual
        cosa la comentarem i la descomentaremos quan tinguem creat l'arxiu .env
        before('deploy:symlink', 'artisan:migrate');
        // Podem definir tasques addicionals que volem que s'executen quan duguem a terme un deploy de
```

l'aplicació i que veurem en punts posteriors

Cadascuna d'aquestes tasques anterior, poden ser executades individualment des del terminal, anteposant al seu nom la paraula dep.

```
// Desplega l'aplicació executant totes les tasques definides en task ('deploy'...
$ dep deploy
// Executa les tasques deploy:info
$ dep deploy:info
```

```
profe@client-desenvolupament$ dep deploy
✈ Deploying màster       192.168.1.118
✔ Executing task deploy:prepare
✔ Executing task deploy:lock
✔ Executing task deploy:release
➤ Executing task deploy:update_code
Cloning into '/var/www/ddaw-ud4-a4/html/releases/6'...
remalnom: Enumerating objects: 114, done.
remalnom: Counting objects: 100% (114/114), done.
remalnom: Compressing objects: 100% (85/85), done.
remalnom: Total 114 (delta 11), reused 114 (delta 11)
Receiving objects: 100% (114/114), 58.75 KiB | 294.00 KiB/s, done.
Resolving deltes: 100% (11/11), done.
Counting objects: 114, done.
Compressing objects: 100% (85/85), done.
Writing objects: 100% (114/114), done.
Total 114 (delta 11), reused 114 (delta 11)
Connection to 192.168.1.118 closed.
✔ Ok
✔ Executing task deploy:shared
✔ Executing task deploy:vendors
✔ Executing task deploy:writable
✔ Executing task artisan:storage:link
✔ Executing task artisan:view:cache
✔ Executing task artisan:config:cache
  Executing task artisan:optimize
```

3.4 Estructura de fitxers i directoris de desplegament

Després d'executar el desplegament de l'aplicació **Deployer** generarà automàticament una sèrie de carpetes i fitxers dins del directori arrel del servidor configurat com a directori "de desplegament" en el nostre cas `/var/www/ddaw-ud4-a4/html/` . A continuació es

detalla l'estructura i la funció que realitza cadascun d'ells.

```
├── .dep
│  ├── current -> releases/1
│  ├── releases
│  │  └── 1
│  └── shared
│     ├── .env
│     └── storage
```

- **releases:** conté les diferents versions de la nostra aplicació que s'han desplegat, fins a un màxim de les especificades per la variable `set('keep_releases', 5);`
- **.dep:** conté metadades per al funcionament de l'eina Deployer.
- **shared:** conté aquells fitxers i directoris que han de mantindre's durant els desplegaments. No hem d'oblidar que a cada desplegament es clona el projecte sencer i es crea una nova versió.
- **current:** És un enllaç a l'última versió de l'aplicació publicada. Quan es crea una nova versió i s'han executat totes les tasques correctament, es crearà un nou enllaç des del directori **current** a l'última versió pujada, de manera que el **downtime serà 0.**

El **document root** del **server block** de NGinx ha d'apuntar a l'aplicació continguda en el directori **current**. És a dir, ha d'apuntar a

`/var/www/ddaw-ud4-a4/html/current/public/`

Activitat 3

Llig la documentació oficial de Deployer i contesta les següents preguntes

- Com podem fer perquè quan executem un **deploy** tingam més informació del que s'està duent a terme en cada pas **(Verbosity)**?
- Quin comando ens permet tornar a una versió desplegada anteriorment?

3.4.1 Arxiu .env

El següent pas serà la creació del fitxer .env amb les variables d'entorn del nostre projecte. Només haurem de generar l'arxiu **.env** durant el primer desplegament, ja que es mantindrà per als **successius desplegaments.**

Ens **connectarem al servidor** amb l'usuari de desplegament ddaw-ud4-deployer i crearem l'arxiu .env al path /var/www/ddaw-ud4-a4/html/shared/.env.

Amb tots els **projectes Laravel** es facilita **una plantilla** un arxiu exemple, amb el nom .env.example de manera que només haurem de dur a terme una còpia.

```
$ sudo cp
/var/www/ddaw-ud4-a4/html/shared/.env.example
/var/www/ddaw-ud4-a4/html/shared/.env
```

A continuació, **modificarem** cadascuna de les **variables** presents en l'arxiu (tal com vam veure en la pràctica anterior) amb els **valors correctes**.

A continuació exemplificarem la modificació de l'arxiu .env per a un **entorn de producció**, per la qual cosa els **logs es desactivaran.** Per a poder veure els diferents problemes que sorgisquen durant la pràctica **es recomana activar els** logs com si es tractara d'un servidor de desenvolupament.

Si una vegada hem creat el fitxer de .env vegem que **quan accedim a l'aplicació**, **el nostre projecte no fa ús d'ell**, necessitarem r**egenerar la caché** de laravel executant la següent ordre desde el directori arrel del projecte.

```
$ artisan config:cache
```

/var/www/html/laravel-app/shared/.env

```
APP_NAME=nomProjecte
APP_ENV=production
```

```
APP_KEY=base64:cA1hATAgR4BjdHJqI8aOj8jEjaaOM8gMNHXIP8d5IQg=
APP_DEBUG=false
APP_LOG_LEVEL=error
APP_URL=http://example.com

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=project_database
DB_USERNAME=project_db_user
DB_PASSWORD=password
....
```

Una altra opció, seria **crear un arxiu** .env.production en la nostra **màquina local** i **copiar-ho** al servidor mitjançant la definició **una nova tasca que afegirem al final del fitxer.**

```
task('upload:env', function () {
    upload('.env.production', '{{deploy_path}}/shared/.env');
})->desc('Environment setup');
```

Per a executar aquesta tasca executem el comando 'dep upload:env'. **Deployer** buscarà l'arxiu .env.production i tractarà de copiar-ho en el path /var/www/html/laravel-app/shared/shared/.env.

3.5 Tasques addicionals. El Servidor php-fpm

El **document root** del server block d'apatxe és un enllaç simbòlic a l'última versió publicada en el directori `current → releases`. Aquesta acció genera un xicotet inconvenient en el nostre servidor d'aplicacions, a causa de l'ús de la cache (**opcache)** de l'intèrpret **php.**

OPcache millora el rendiment de PHP emmagatzemant el codi precompilat (**bytecode)** dels diferents **scripts** de l'aplicació en la memòria compartida, **eliminant** així la necessitat de que carregue els scripts en **cada petició.**

No obstant això, al modificar-se només el destí de l'enllaç simbòlic en cadascun dels desplegaments, el servidor d'aplicacions no detectarà cap canvi en el codi font i continuarà utilitzant aquells scripts precompilats que manté en caché. Per assegurar-nos que els canvis de desplegament són aplicats, haurem de reiniciar el servei **php-fpm** al final del procés.

```
$ sudo /etc/init.d/php8.1-fpm restart
```

Aquesta acció es durà a terme a cada desplegament, per la qual cosa haurem d'automatitzar-la i incloure-la com l'última tasca del procés de desplegament. Necessitarem:

- Que l'usuari `ddaw-ud4-deployer` tinga **suficients privilegis** per a reiniciar el servei **php-fpm**.
- Crear una **nova tasca** el fitxer de `deploy.php` i registrar-la perquè es duga a terme just quan finalitze el procés de desplegament (veurem com fer-ho als següents punts)

3.5.1 Otorgar privilegis a l'usuari. Arxiu sudoer.

L'arxiu **sudoers** conté una llista dels usuaris que poden executar el comando **sudo** i quins són els abastos dels seus **privilegis**. Quan un usuari executa un comando precedit per sudo, el sistema busca en l'arxiu /etc/sudoers i els arxius situats en /etc/sudoers.d per a comprovar la informació allí indicada i **atorgar o denegar** el permís per a executar **el comandament**. Les directives presents en aquest fitxer tenen la forma:

```
qui on = (com_qui) què
```

Un possible exemple de directiva és el següent:

```
# L'usuari root en tots els hosts pot executar qualsevol comando com qualsevol usuari
root            ALL=(ALL)            ALL
```

Editarem el fitxer /etc/sudoers i inclourem la següent directiva. La forma **recomanada i segura** d'editar l'arxiu és utilitzant el comandament visudo.

```
$ sudo visudo
```

/etc/sudoers
ddaw-ud4-deployer ALL=(ALL) NOPASSWD: /etc/init.d/php8.1-fpm restart

Una vegada guardats els canvis, podrem reiniciar el servei sense problemes utilitzant l'usuari de desplegament.

```
ddaw-ud4-deployer@server:~$ sudo /etc/init.d/php8.1-fpm restart
[ ok ] Restarting php8.1-fpm (via systemctl): php8.1-fpm.service
```

Activitat 4

Per què és més segur la utilització de l'eina visudo per a l'edició del fitxer /etc/sudoers?

Com podríem crear un àlies a l'arxiu sudoers, que incloguera els permisos per al reinici del **servei NGinx** i del **servei php-fpm**? Consulta la documentació oficial. (No és necessari que ho implementes, només contesta la pregunta)

3.5.2 Afegir la nova tasca al cicle de desplegament.

Per a finalitzar declararem una **nova tasca** amb l'identificador reload:php-fpm en l'arxiu **deploy.php** que ens permeta reiniciar el servei i la inclourem en el cicle de desplegament de l'aplicació. D'aquesta manera sempre que s'execute el comando dep deploy, i finalitze amb èxit, es reiniciarà el servei **php-fpm.**

```
# Declaració de la tasca
task('reload:php-fpm', function () {
    run('sudo /etc/init.d/php8.1-fpm restart');
});
# inclusió en el cicle de desplegament
after('deploy', 'reload:php-fpm');
```

4. Flux de desplegament.

El cicle de desenvolupament i **desplegament diari** de l'aplicació a partir de l'arquitectura configurada quedarà definit per **la realització dels següents passos des de la màquina**

local on es troba l'entorn de desenvolupament.

1. Realització dels canvis en el codi font del projecte.
2. Creació del **commit** amb els canvis realitzats i pujada **push** al repositori central de gitlab/github.

```
$ git commit -am '[*ADD] Feature a43534sg – CRUD usuaris'
$ git push origin master
```

3. Execució de la tasca deploy configurada en **Deployer**.

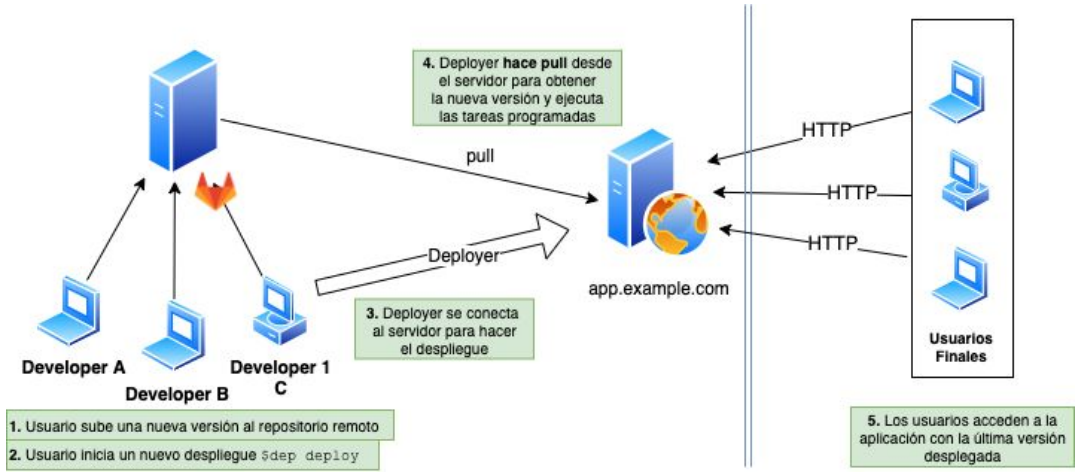
```
$ dep deploy
```

4. Deployer es connectarà al servidor utilitzant la clau privada configurada al fitxer deploy.php.

5. Una **vegada connectat a la màquina remota,** clonarà el repositori al servidor fent ús de l'usuari amb el qual estiguem connectats al servidor (en el nostre exemple anterior ddaw-ud4-deployer) i executarà les tasques programades.

6. Una vegada finalitzades i executades **totes les tasques correctament**, els usuaris tindran disponible la nova versió de l'aplicació.

A continuació es presenta un diagrama explicatiu de l'arquitectura i el flux de desplegament proposat:



5. Treball a realitzar

Activitat 5

Grava un vídeo, al que deveu mostrar i explicar la configuració l'entorn de desplegament de l'aplicació **Laravel** que estàs desenvolupant en l'assignatura de programació web en entorn servidor utilitzant l'eina **Deployer.** Hauràs de tindre en compte els següents requisits:

- Hauràs de configurar un **server block** específic per a l'aplicació.
- El directori de desplegament serà: /var/www/prod-ud4-a4/html.
- L'usuari de desplegament serà: prod-ud4-deployer.

6. Referències

- Digital Ocean. Automatically deploy laravel application on Ubuntu. "https://www.digitalocean.com/community/tutorials/automatically-deploy-laravel-applications-deployer-ubuntu".
- Laravel Framework. Documentació oficial. "https://laravel.com/docs/6.x/configuration"
- Deployer. Documentació oficial. "https://deployer.org/docs/getting-started.html"
- https://www.sudo.ws/man/1.8.17/sudoers.man.html