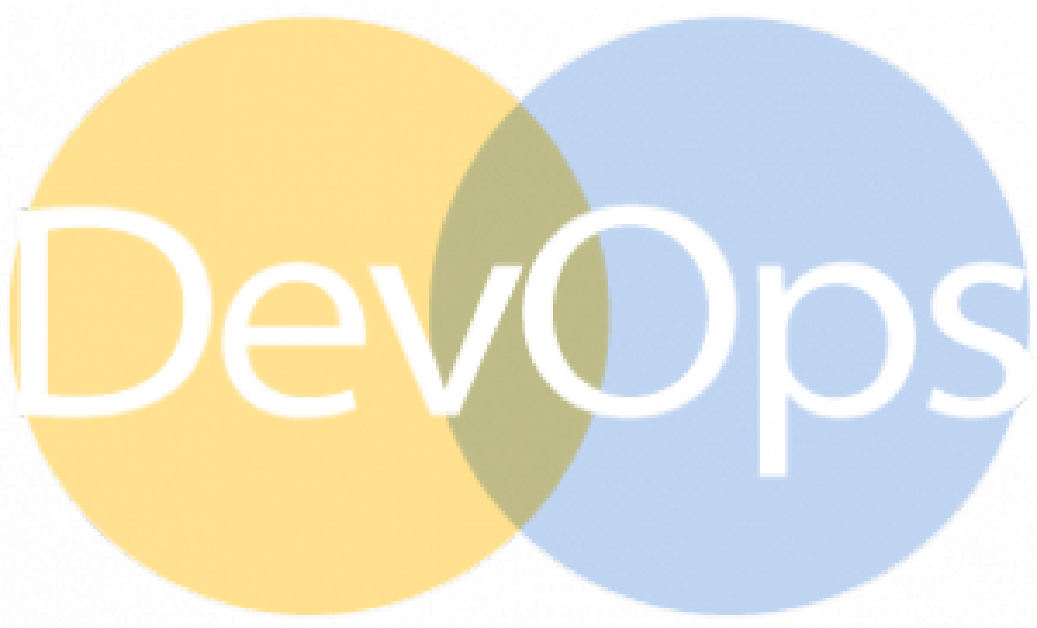
МАСШТАБИРУЕМЫЙ DEVOPS

АЛЕКСАНДР КОЛЕСЕНЬ

О СЕБЕ

- **shop.by**
  - System Engineer
  - FreeBSD, Linux, Apache, nginx, C, C++, perl, MySQL
- **Wargaming.net**
  - Web DevOps
  - Linux, Apache, nginx, AMQP/RabbitMQ, Python, Django, MySQL, memcached, lxc, puppet, fabric
- **SiliconMint, Iron.io**
  - DevOps, System Engineer
  - Linux, Python, Ruby, RoR, Go, MongoDB, MySQL, Redis, memcached, AMQP/RabbitMQ, AWS, lxc, chef, fabric, Capistrano


DevOps

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

- FreeBSD vs. Ubuntu vs. CentOS vs. Debian vs. ...

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

- FreeBSD vs. Ubuntu vs. CentOS vs. Debian vs. ...
- apache vs. uwsgi vs. tornado vs. ...

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

- FreeBSD vs. Ubuntu vs. CentOS vs. Debian vs. ...
- apache vs. uwsgi vs. tornado vs. ...
- MySQL vs. PgSQL vs. CouchBase vs. MongoDB vs. ...

ПЕРВЫЕ ШАГИ

```
# make
# cp libproject.so /var/www/
```

или

```
$ scp -r site host:/tmp/
# cp -R /tmp/site /var/www/

# apachectl restart
...
# killall -9 httpd
# apachectl start
```

VCS UP?!

```
# apachectl stop
# svn up / git pull
# apachectl start
```

VCS UP?!

```
# apachectl stop
# svn up / git pull
# apachectl start

# svn / git log
...
"Last fix on #456"
"Still fixing #456"
"One more commit on #456"
"Fixed bug #456"
```

ФАЗЫ ДЕПЛОЙМЕНТА

- подготовка окружения
  - основная работа выполняется единоразово
  - инкрементальные обновления (конфиги, софт и т.п.)

ФАЗЫ ДЕПЛОЙМЕНТА

- подготовка окружения
  - основная работа выполняется единоразово
  - инкрементальные обновления (конфиги, софт и т.п.)
- деплоймент
  - полностью каждый раз при обновлении

ОКРУЖЕНИЕ

ОКРУЖЕНИЕ

- смотрим на продакшн

ОКРУЖЕНИЕ

- смотрим на продакшн
- руками :(ставим такие же пакеты

ОКРУЖЕНИЕ

- смотрим на продакшн
- руками :(ставим такие же пакеты
- пакета нет - ./configure && make && make install :(

ОКРУЖЕНИЕ

- смотрим на продакшн
- руками :(ставим такие же пакеты
- пакета нет - ./configure && make && make install :(
- (может быть) записываем действия в .sh-скрипт

ОКРУЖЕНИЕ

- смотрим на продакшн
- руками :(ставим такие же пакеты
- пакета нет - ./configure && make && make install :(
- (может быть) записываем действия в .sh-скрипт

Итог:

- Ubuntu, CentOS etc. превращается в “Slackware”

ОКРУЖЕНИЕ

- смотрим на продакшн
- руками :(ставим такие же пакеты
- пакета нет - ./configure && make && make install :(
- (может быть) записываем действия в .sh-скрипт

Итог:

- Ubuntu, CentOS etc. превращается в “Slackware”
- много чего забывается

ОКРУЖЕНИЕ

- смотрим на продакшн
- руками :(ставим такие же пакеты
- пакета нет - ./configure && make && make install :(
- (может быть) записываем действия в .sh-скрипт

Итог:

- Ubuntu, CentOS etc. превращается в “Slackware”
- много чего забывается
- конфигурация скорей всего будет различаться

ОКРУЖЕНИЕ: АВТОМАТИЗАЦИЯ

Декларативная конфигурация

Важно, ЧТО будет в итоге, а не КАК это будет выполнено

ОКРУЖЕНИЕ: АВТОМАТИЗАЦИЯ

Chef, puppet

```
cron "mycrontab" do
  minute "0"
  hour "0,12"
  user "www-rest"
  command "cd /home/#{app_config[:user]}/app/src/worker
           && . ../../bin/activate
           && python db_validator.py
           ../../conf/app.json"
  mailto "#{app_config[:email]}"
end
```

ОКРУЖЕНИЕ: АВТОМАТИЗАЦИЯ

Запуск

```
$ knife ssh 'name:api.myservice.com' 'sudo chef-client'
```

Не важно, как, но запись оказывается в crontab:

```
$ crontab -l
# Chef Name: mycrontab
MAILTO=alexander.kolesen@gmail.com
0 0,12 * * * cd /home/www-rest/app/src/worker
              && . ../../bin/activate
              && python db_validator.py ../../conf/app.json
```

ДЕПЛОЙМЕНТ: АВТОМАТИЗАЦИЯ

Императивный процесс

1. закачиваем и распаковываем исходники
2. закрываем доступ к серверу
3. останавливаем сервер
4. обновляем исходный код
5. накатываем миграции
6. перезапускам фоновые рабочие задачи
7. запускаем сервер
8. тестируем
9. открываем доступ к серверу

ДЕПЛОЙМЕНТ: АВТОМАТИЗАЦИЯ

Fabric, capistrano

```
def upload():
    rsync_project("app/src",
                  "rest worker install tools",
                  exclude = ["*.pyc"],
                  delete=True)

def restart():
    run("nohup %s/app/src/tools/worerks.sh restart  -n8" %
        homedir())
    run("sudo /etc/init.d/apache2 graceful")


def deploy():
    upload()
    restart()
```

ДЕПЛОЙМЕНТ: АВТОМАТИЗАЦИЯ

Запуск

```
$ fab deploy -H api.myservice.com -u www-rest
```

Результат

1. rsync проекта на api.myservice.com
2. перезапуск сервисов

БЕСШОВНОЕ ОБНОВЛЕНИЕ

Это хорошо!

БЕСШОВНОЕ ОБНОВЛЕНИЕ

Это хорошо!

- если обновляются только css/js - нет смысла выключать

БЕСШОВНОЕ ОБНОВЛЕНИЕ

Это хорошо!

- если обновляются только css/js - нет смысла выключать
- если обновляется код - отключаем бэкенды “по очереди”

БЕСШОВНОЕ ОБНОВЛЕНИЕ

Это хорошо!

- если обновляются только css/js - нет смысла выключать
- если обновляется код - отключаем бэкенды “по очереди”
- если обновляется база (миграции) - ReadOnly-режим

БЕСШОВНОЕ ОБНОВЛЕНИЕ

Это хорошо!

- если обновляются только css/js - нет смысла выключать
- если обновляется код - отключаем бэкенды “по очереди”
- если обновляется база (миграции) - ReadOnly-режим
- иначе показываем “картинку”: “Мы обновляемся”

БЕСШОВНОЕ ОБНОВЛЕНИЕ

Nginx, maintenance mode

```
map $remote_addr $under_construction {
    default under_construction.jpg;
    193.232.92.23 .;
}

location / {
    try_files $under_construction @backend;
}

location @backend {
    proxy_pass http://<upstream>;
}
```

СТЕЙДЖИНГ

Пре-продакшн

СТЕЙДЖИНГ

Пре-продакшн

- как можно более близкий по конфигурации к продакшну

СТЕЙДЖИНГ

Пре-продакшн

- как можно более близкий по конфигурации к продакшну
- но без фанатизма

СТЕЙДЖИНГ

Пре-продакшн

- как можно более близкий по конфигурации к продакшну
- но без фанатизма
- вечный дефицит железа для стейджингов

СТЕЙДЖИНГ

Пре-продакшн

- как можно более близкий по конфигурации к продакшну
- но без фанатизма
- вечный дефицит железа для стейджингов
- поэтому виртуальные окружения: lxc, kvm, xen - OK

СТЕЙДЖИНГ

Пре-продакшн

- как можно более близкий по конфигурации к продакшну
- но без фанатизма
- вечный дефицит железа для стейджингов
- поэтому виртуальные окружения: lxc, kvm, xen - OK
- если мы на AWS/EC2, Rackspace - не нужно железа, OK

ТЮРЬМА

ТЮРЬМА

- приложение работает от “restricted” пользователя

ТЮРЬМА

- приложение работает от “restricted” пользователя
- максимум - пару sudo на graceful-перезапуск Web-сервера

ТЮРЬМА

- приложение работает от “restricted” пользователя
- максимум - пару sudo на graceful-перезапуск Web-сервера
- приложение не навредит системе

ТЮРЬМА

- приложение работает от “restricted” пользователя
- максимум - пару sudo на graceful-перезапуск Web-сервера
- приложение не навредит системе
- ни другим приложениям

ТЮРЬМА

- приложение работает от “restricted” пользователя
- максимум - пару sudo на graceful-перезапуск Web-сервера
- приложение не навредит системе
- ни другим приложениям
- пользователь (разработчик) тоже

ТЮРЬМА

- приложение работает от “restricted” пользователя
- максимум - пару sudo на graceful-перезапуск Web-сервера
- приложение не навредит системе
- ни другим приложениям
- пользователь (разработчик) тоже
- можно разрешать деплоить самому разработчику

ТЮРЬМА

- приложение работает от “restricted” пользователя
- максимум - пару sudo на graceful-перезапуск Web-сервера
- приложение не навредит системе
- ни другим приложениям
- пользователь (разработчик) тоже
- можно разрешать деплоить самому разработчику
- root-только для диагностики, в крайних случаях

ТЮРЬМА

- приложение работает от “restricted” пользователя
- максимум - пару sudo на graceful-перезапуск Web-сервера
- приложение не навредит системе
- ни другим приложениям
- пользователь (разработчик) тоже
- можно разрешать деплоить самому разработчику
- root-только для диагностики, в крайних случаях
- все остальное - chef, puppet

SSH-КЛЮЧИ

SSH-КЛЮЧИ

- private & public-части

SSH-КЛЮЧИ

- private & public-части
- private - ТОЛЬКО у вас

SSH-КЛЮЧИ

- private & public-части
- private - ТОЛЬКО у вас
- НЕЛЬЗЯ раздавать по скайпу, почте, на флешке

SSH-КЛЮЧИ

- private & public-части
- private - ТОЛЬКО у вас
- НЕЛЬЗЯ раздавать по скайпу, почте, на флешке
- public - на сервере, в authorized_keys

SSH-КЛЮЧИ

- private & public-части
- private - ТОЛЬКО у вас
- НЕЛЬЗЯ раздавать по скайпу, почте, на флешке
- public - на сервере, в authorized_keys
- если кому-то нужен доступ - добавляем в authorized_keys

SSH-КЛЮЧИ

- private & public-части
- private - ТОЛЬКО у вас
- НЕЛЬЗЯ раздавать по скайпу, почте, на флешке
- public - на сервере, в authorized_keys
- если кому-то нужен доступ - добавляем в authorized_keys
- с помощью chef/puppet

CONTINIOUS INTEGRATION

Как web frontend к скриптам деплоймента

CONTINIOUS INTEGRATION

Как web frontend к скриптам деплоймента

- деплоймент “по кнопке”

CONTINIOUS INTEGRATION

Как web frontend к скриптам деплоймента

- деплоймент “по кнопке”
- деплоймент “по коммиту”

CONTINIOUS INTEGRATION

Как web frontend к скриптам деплоймента

- деплоймент “по кнопке”
- деплоймент “по коммиту”
- деплоймент “по крону”, раз в час/два/день

ИТОГО

- использовать проверенный софт
- автоматизировать подготовку окружения
- автоматизировать деплоймент приложения
- сделать обновление максимально “бесшовным”
- деплоить на стейджингах, потом - в продакшне
- ограничивать себя в правах
- дать пользователям кнопку “сделать хорошо”

СПАСИБО ЗА ВНИМАНИЕ. ВОПРОСЫ

Александр Колесень

mailto:alexander.kolesen@gmail.com

https://twitter.com/imm0use

https://plus.google.com/107935551373006842102/