

# **DevOps avec Ansible et Docker**

**le 20/03/2015**

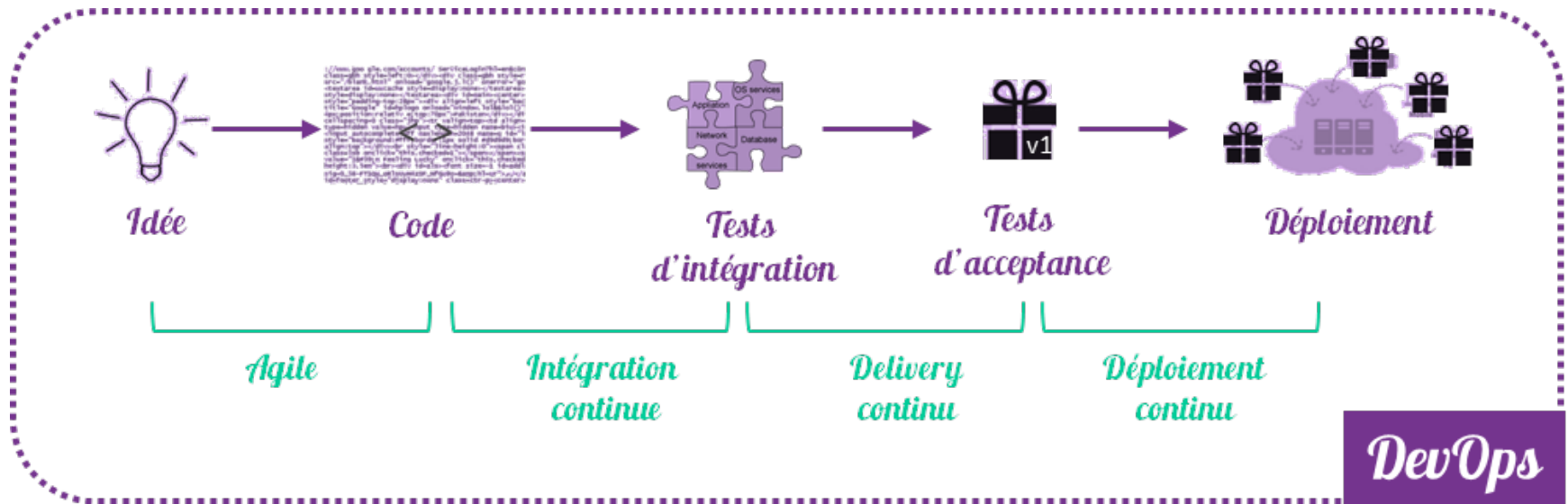
**Stéphane Manciot**

# Présentation

- Introduction à DevOps
- Packaging avec Docker
- Pourquoi Ansible ?
- Déploiement et provisioning avec Ansible



# DevOps



# DevOps - Bénéfices attendus (1/5)



## **Des cycles de déploiement plus courts**

Les devOps jouent un rôle clé dans la réduction du temps du cycle de déploiement des logiciels, passant de quelques semaines à seulement quelques heures, permettant une plus grande flexibilité quant aux nouvelles fonctionnalités et changements à apporter au produit initial.



# DevOps - Bénéfices attendus (2/5)



## Mise à disposition de nouveaux services plus rapidement

Des déploiements fréquents associés à des délais de livraison plus rapides permettent une **agilité opérationnelle**.

# DevOps - Bénéfices attendus (3/5)



## **Une satisfaction client améliorée**

Grâce à des applications ciblées et de qualité, conformes aux retours clients end to end.



# DevOps - Bénéfices attendus (4/5)



## **Des coûts réduits**

L'automatisation permet aux équipes de réaffecter des ressources précieuses à des tâches à plus haute valeur.

# DevOps - Bénéfices attendus (5/5)

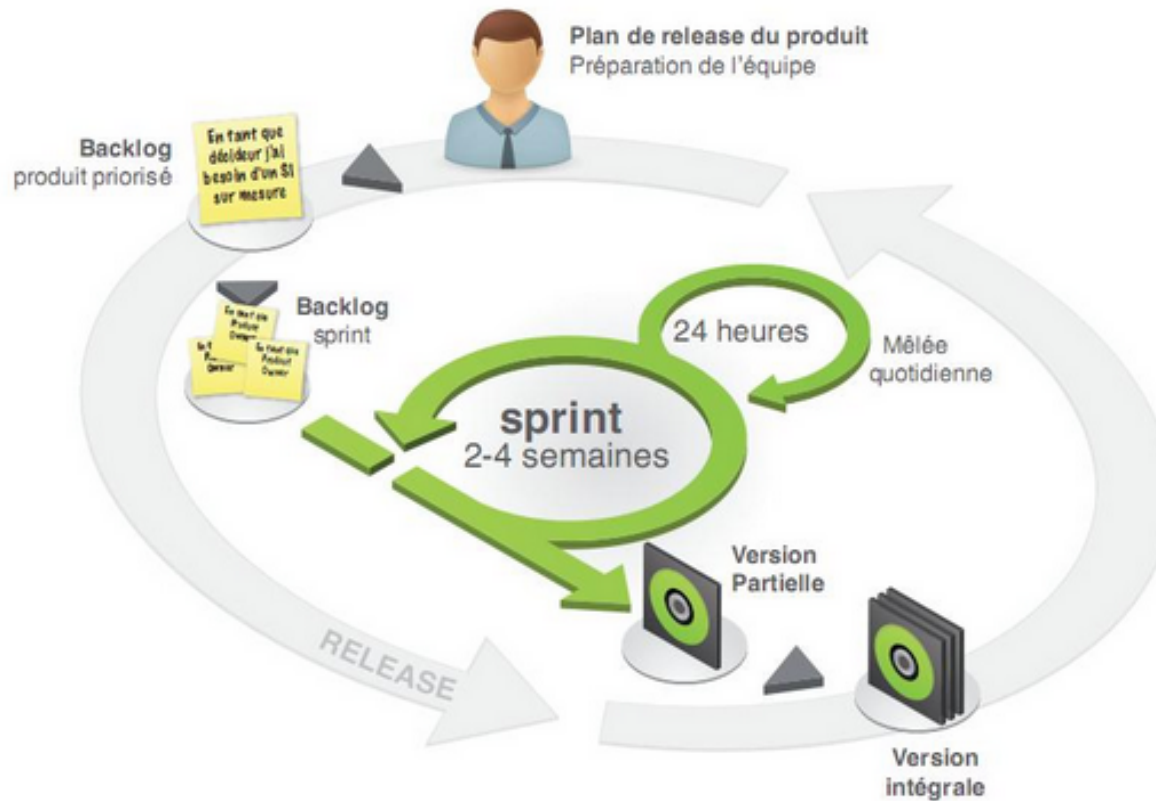


## **Conformité et Gouvernance**

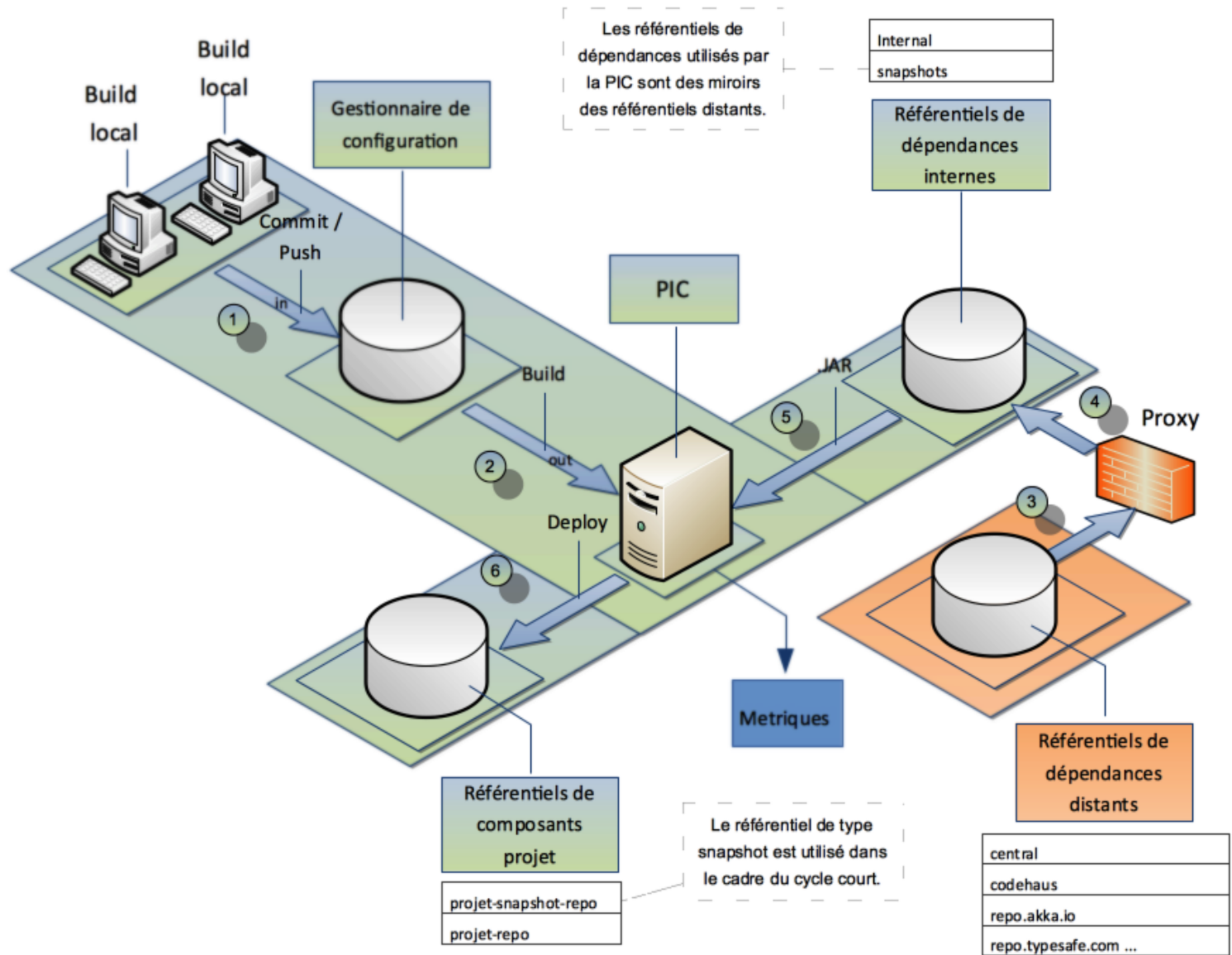
Automatisation du tracking et reporting end-to-end sur les phases de livraison / déploiement continu.



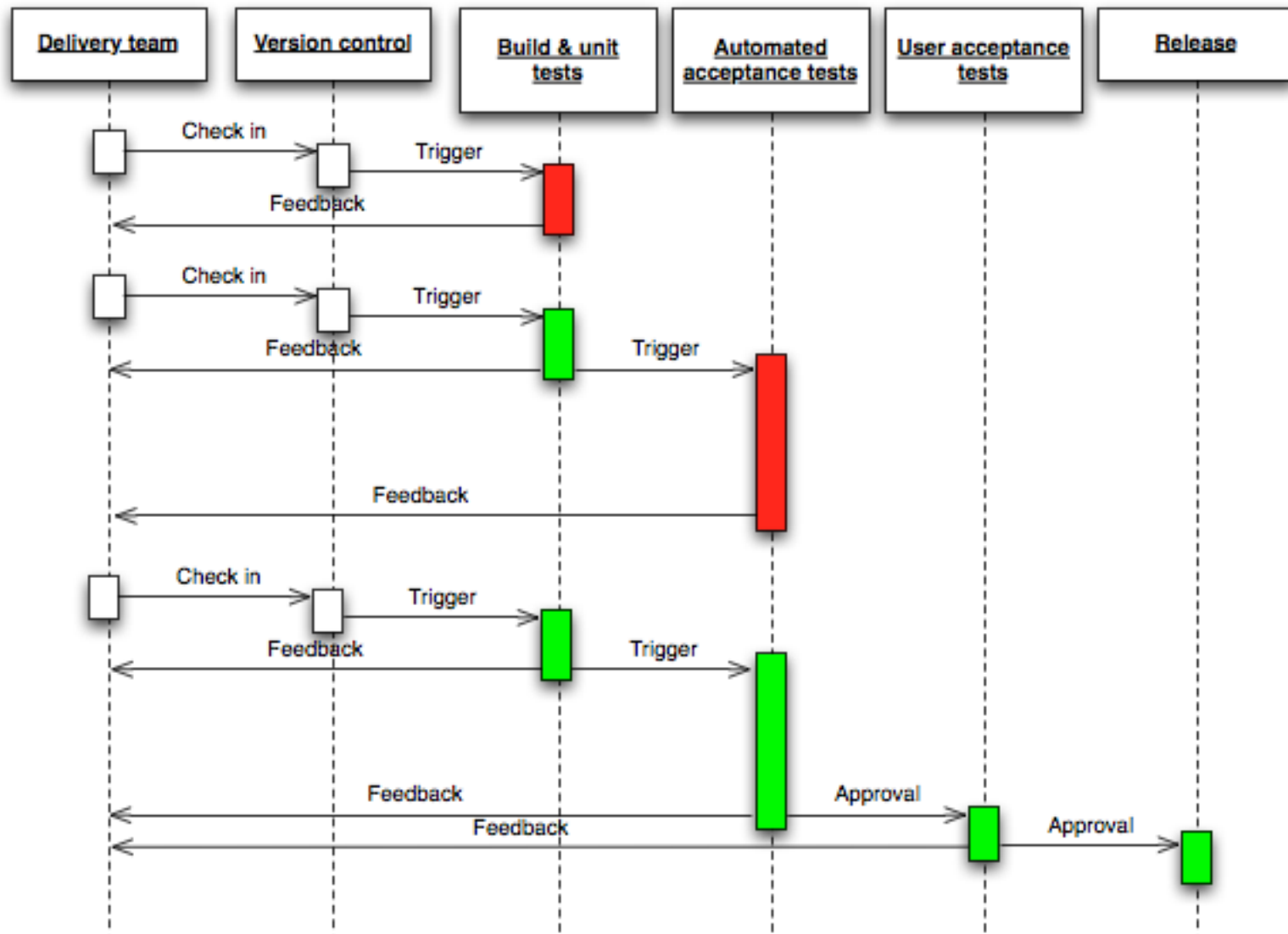
# DevOps - Démarche agile



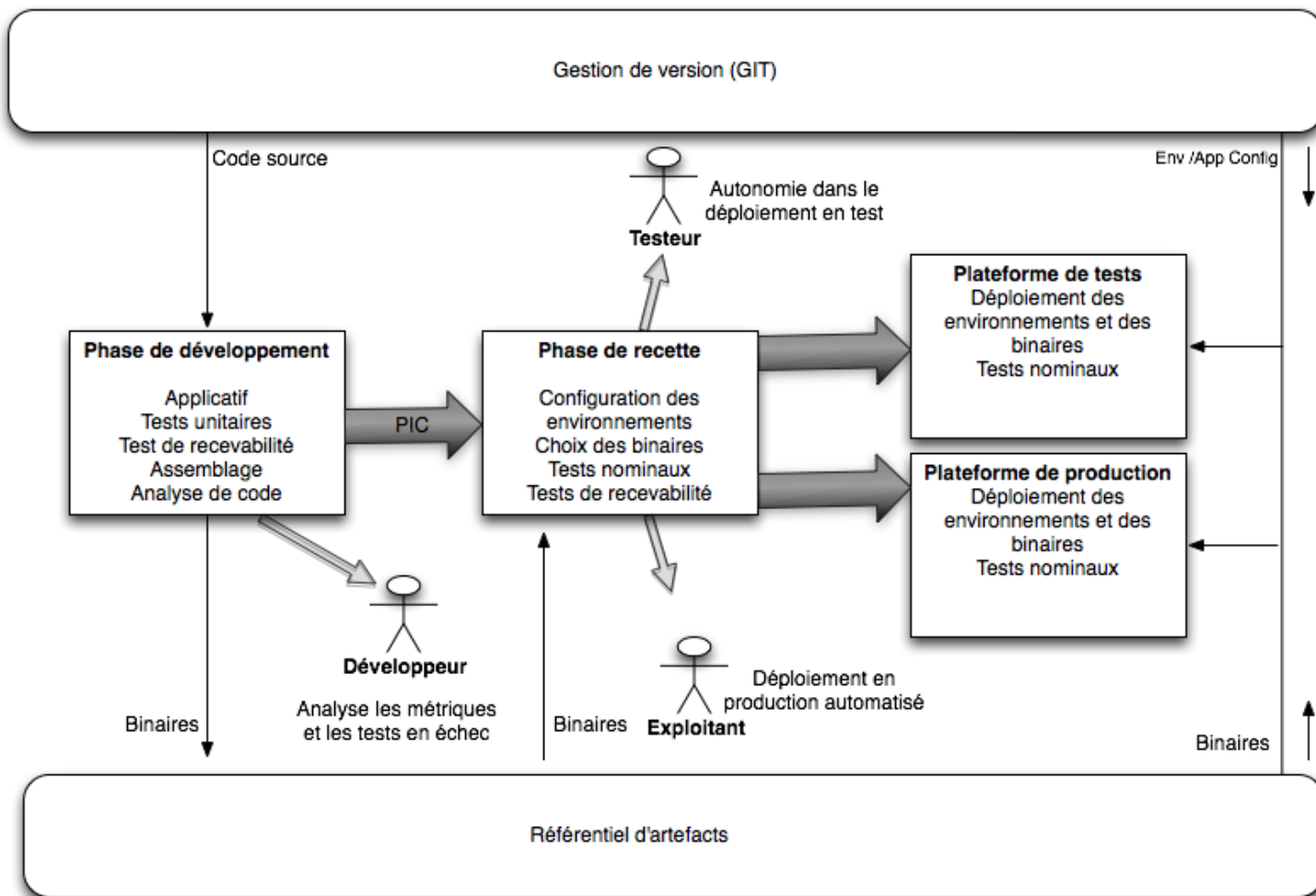
# DevOps - Intégration continue



# DevOps - Livraison continue

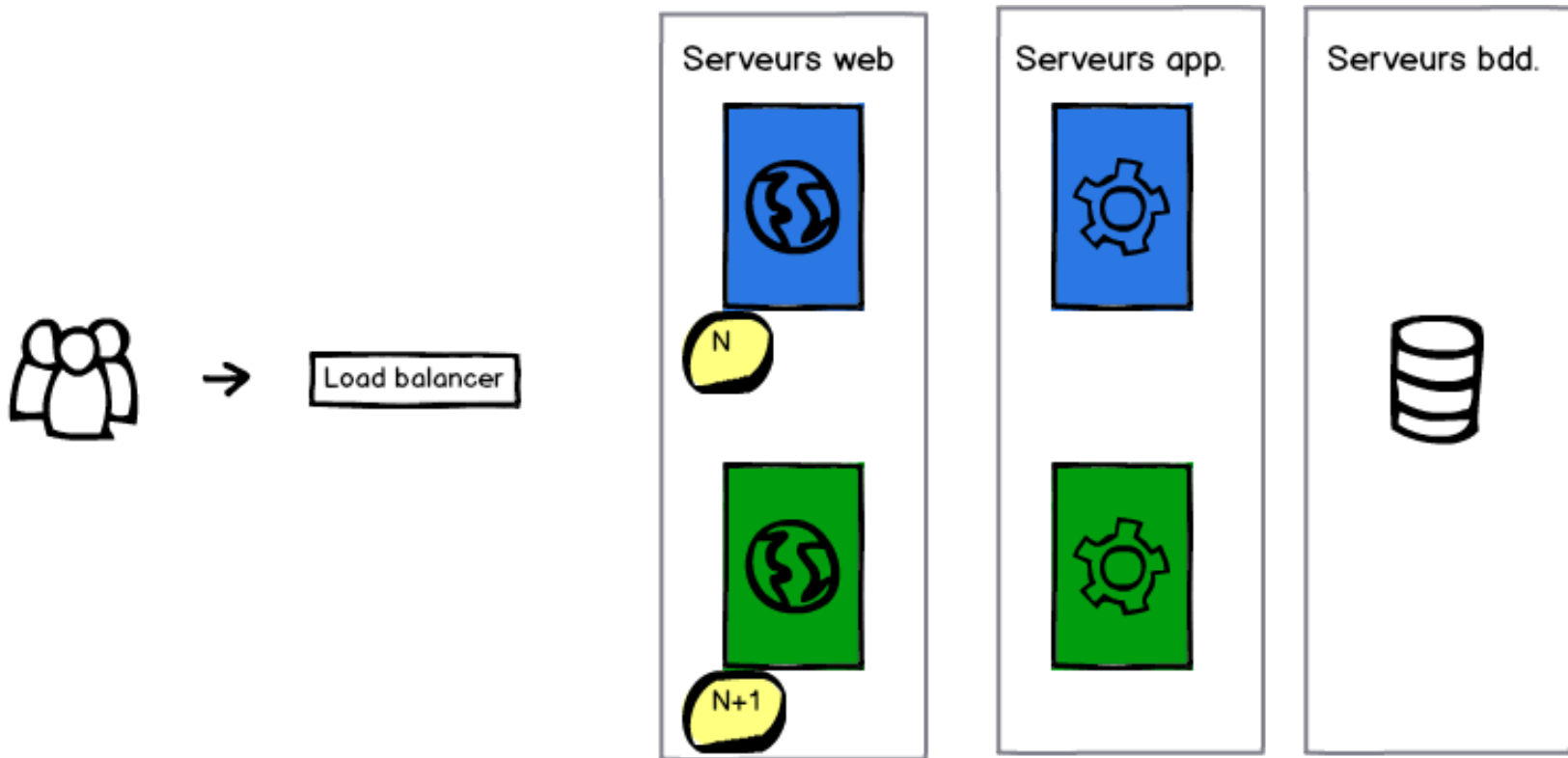


# DevOps - Déploiement continu



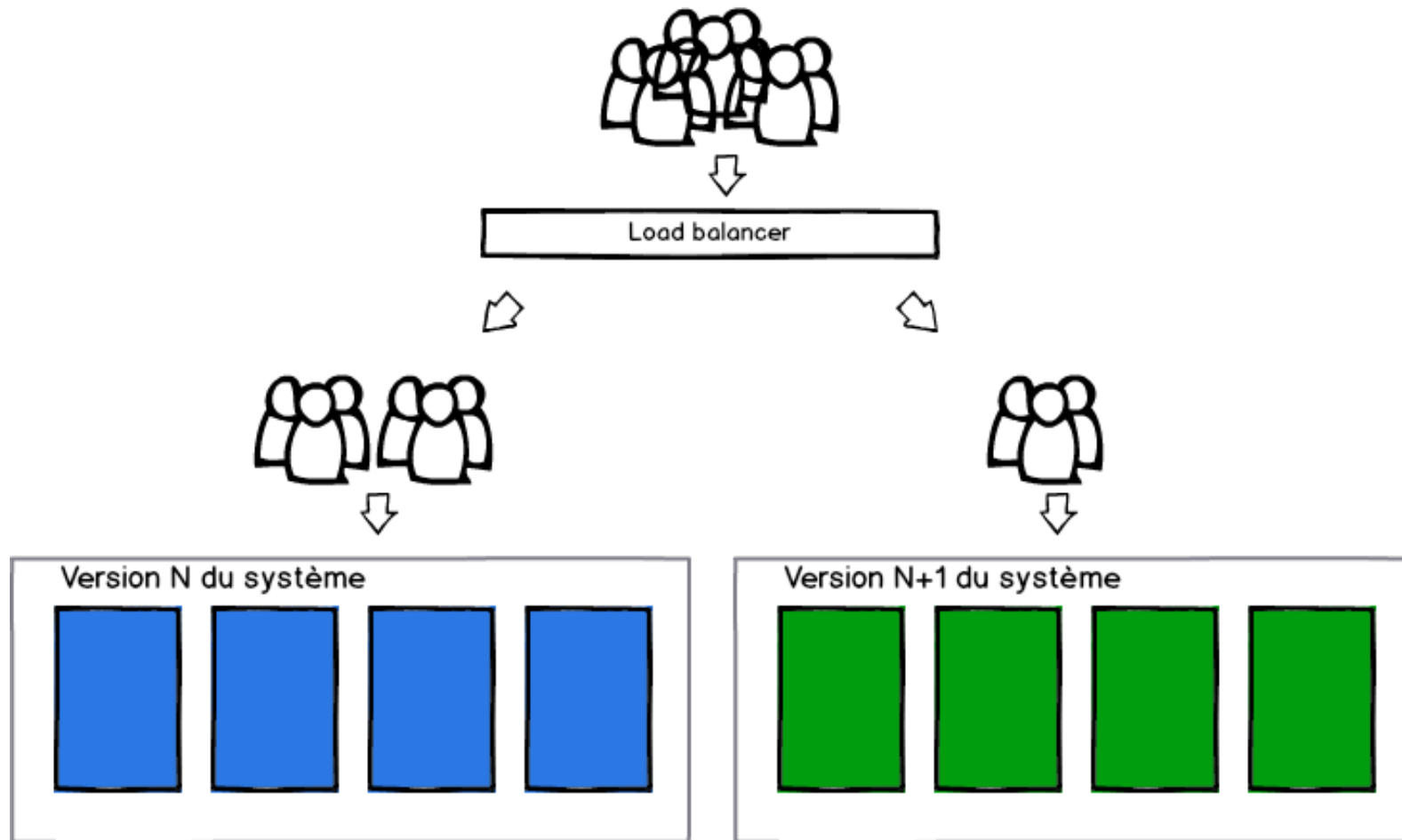
# DevOps - Déploiement Zero Downtime (1/2)

- Pattern Blue/Green



# DevOps - Déploiement Zero Downtime (2/2)

- Pattern Canary







































# DevOps - Problématique

django web frontend	?	?	?	?	?	?
node.js async API	?	?	?	?	?	?
background workers	?	?	?	?	?	?
SQL database	?	?	?	?	?	?
distributed DB, big data	?	?	?	?	?	?
message queue	?	?	?	?	?	?
	my laptop	your laptop	QA	staging	prod on cloud VM	prod on bare metal



# Docker - le conteneur intermodal



django web frontend						
node.js async API						
background workers						
SQL database						
distributed DB, big data						
message queue						
	my laptop	your laptop	QA	staging	prod on cloud VM	prod on bare metal



# Docker - DevOps



- Construction rapide, idempotent et automatique d'images pouvant être partagées (Dockerfile, docker-registry ...)
- Séparation des rôles
  - Développeur : à la main sur le conteneur
  - Opérationnel : à la main sur le reste
    - gestion des logs
    - gestion des accès distants
    - configuration réseau
    - monitoring
    - ...



# Docker - PaaS



- Portabilité
- Provisioning rapide (Another Union File System)
- Performance : les avantages d'une VM (isolation des processus, interface réseau, ...) sans les inconvénients (processus exécutés au sein de l'hôte, pas d'émulation de périphérique)

# Docker - Dockerfile (Exemple)



```
#
# mogobiz-launcher Dockerfile
#
#
# Pull base image.
FROM dockerfile/java:oracle-java7

ENV MOGOBIZ_HOME /mogobiz
ENV JVM_OPT -Xmx1024M -XX:MaxPermSize=1024m

# add mogobiz libraries
ADD lib/* /mogobiz/lib/

# add mogobiz launchers
ADD bin/* /mogobiz/bin/
RUN chmod +x /mogobiz/bin/*

# Define mountable directories.
VOLUME ["/mogobiz/conf", "/mogobiz/data", "/mogobiz/templates", "/mogopay/import",

# Expose ports.
EXPOSE 8080

# Define default command.
CMD /mogobiz/bin/mogobiz-all
```

*image de base*

*variables d'environnement*

*copie de fichiers*

*exécution de commandes*

*port(s) d'écoute*

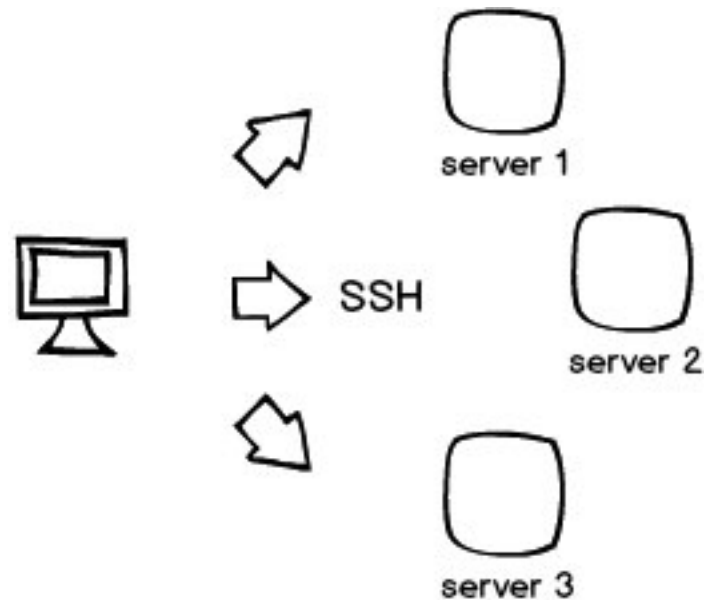
*points de montage*

*commande par défaut*

# Docker - Gestion des images



- rechercher une image : `sudo docker search debian`
- lister les images : `sudo docker images`
- récupérer une image : `sudo docker pull debian`
- exécuter un conteneur : `sudo docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARGS...]`
- lister tous les conteneurs : `sudo docker ps -a`
- récupérer l'id du conteneur lancé en dernier : `sudo docker ps -l`
- commit maj conteneur : `sudo docker commit ID [IMAGE[:TAG]]`
- inspecter un conteneur : `sudo docker inspect ID`
- pousser une image : `sudo docker push IMAGE`



- Orchestration et automatisation des tâches d'administration système
  - provisioning
  - déploiement d'application



# Ansible - Pourquoi ?



- **Simplicité d'exécution** : pas besoin de maître ni d'agent sur les systèmes à administrer (ssh)
- **Mode Push**
- **Simplicité d'apprentissage** (YAML)
- **Performant** : exécution des scripts en parallèle sur les machines cible
- **Extensible** : python
- **DRY** : rôles
- **Idempotent** : chaque tâche est exécutée en garantissant que le système cible sur lequel elle s'applique se trouvera dans l'état désiré post exécution
- **Sécurisé** : ansible-vault



# Ansible - Inventory



*Inventory*  
↑  
ansible **webserver**s -m ping

hosts/preprod :

```
[webserver]
preprod.mon-service.org
```

```
[dbserver]
preprod.mon-service.org
```

```
[dockers:children]
webserver
dbserver
```

hosts/prod :

```
[webserver]
prod.mon-service.org
```

```
[dbserver]
prod.mon-service.org
```

```
[dockers:children]
webserver
dbserver
```

# Ansible - Variables



- **group\_vars/webservers**

---

```
vhost: {servername: "{{servername}}", documentroot: "/www/{{servername}}", serveradmin:
stephane.manciot@ebiznext.com}
```

- **host\_vars/preprod.monservice.org**

---

```
database_name: monservice
database_user: monuser
database_password: monpassword
```





# Ansible - Module



ansible webservers -m ping

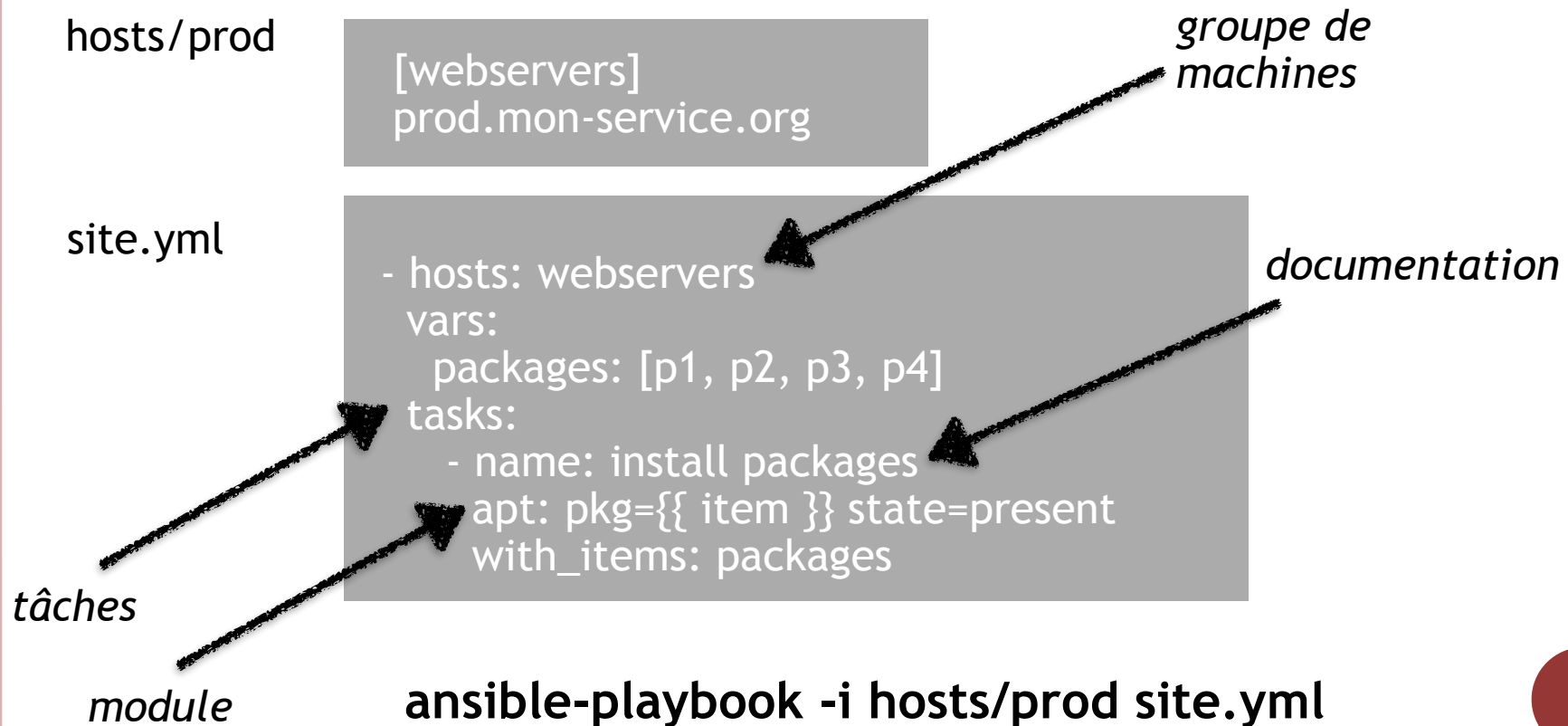


*Module*

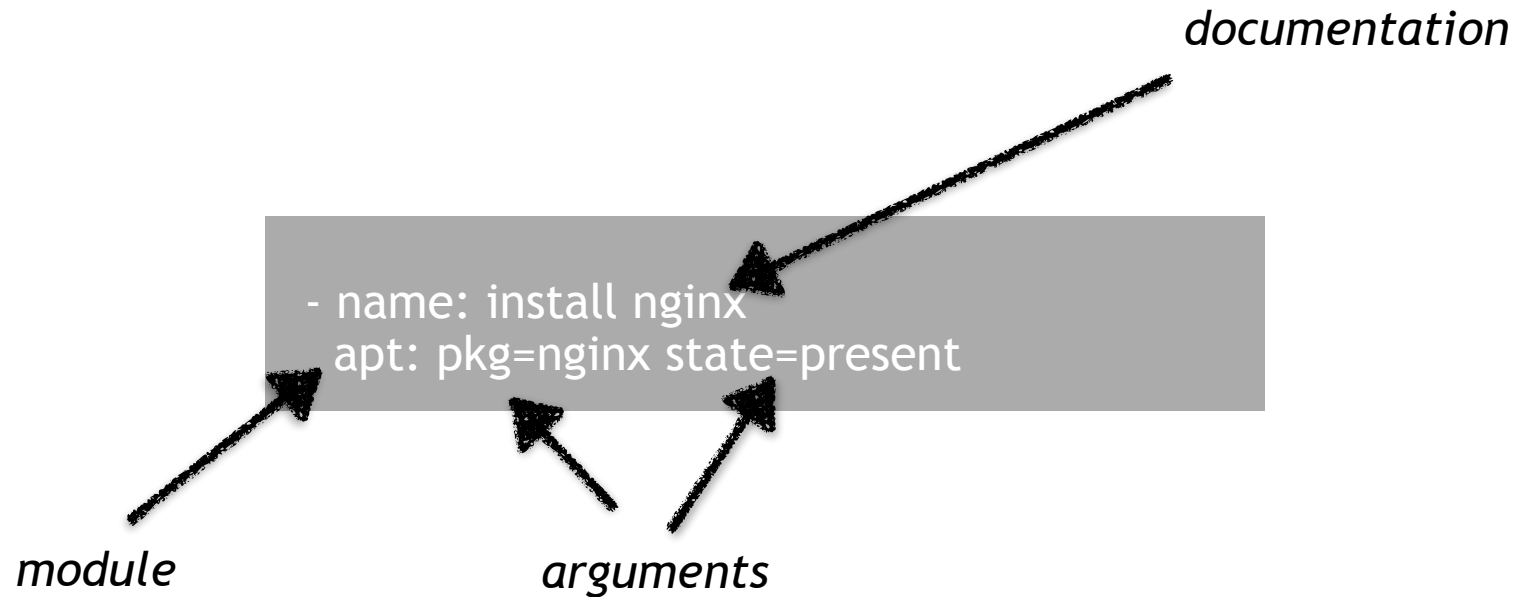
# Ansible - Playbook



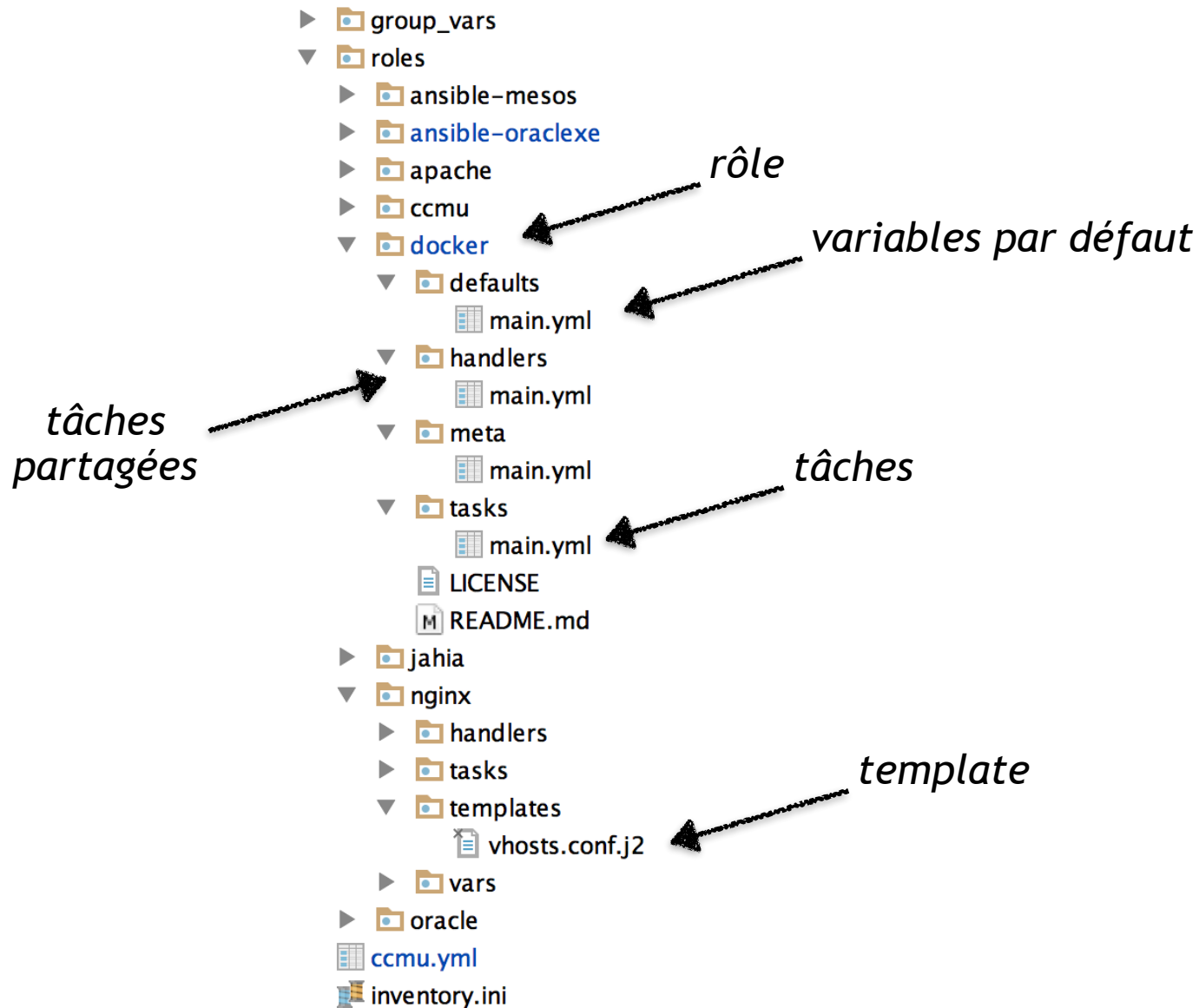
- Exécution de tâches spécifiques sur un ou plusieurs groupes de machines



# Ansible - Playbook / Tâche



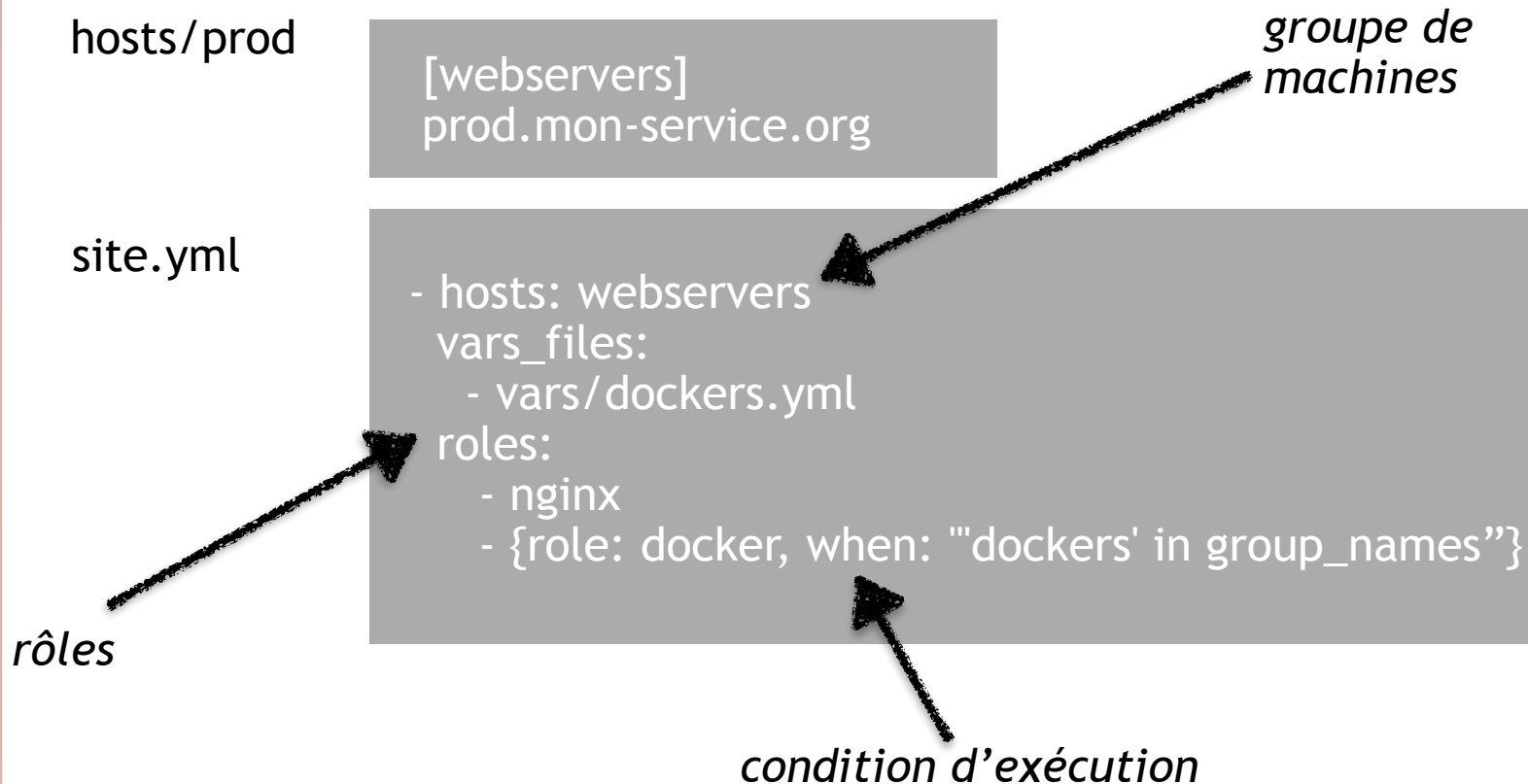
# Ansible - Playbook / Rôle



# Ansible - Playbook / Rôle



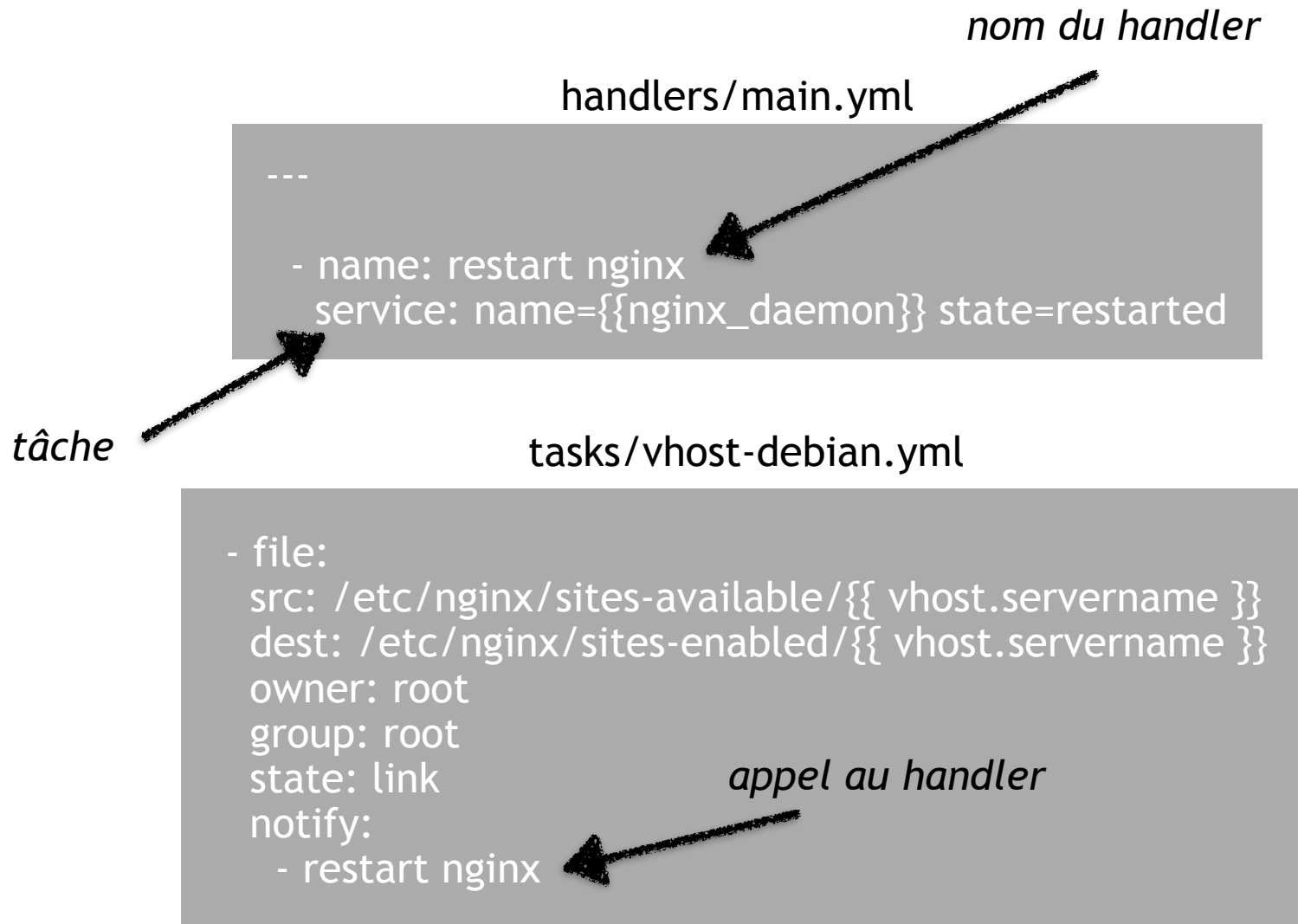
- Exécution de rôles sur un groupe de machines



# Ansible - Playbook / Rôle / Handler



- Exécution de tâche répétitive



# Ansible - Playbook / Rôle / Template



templates/vhosts.conf.j2

```
server {  
    listen 80;  
    server_name {{ vhost.servername }};  
    access_log on;  
    access_log /var/log/nginx/ccmu.ebiznext.com.access.log;  
    location /jahia/ {  
        proxy_pass http://{{jahia_host}}:{{hostvars[jahia_host].jahia_http_port}}/jahia/;  
    }  
    ...  
}
```

*variable*

tasks/vhost-debian.yml

```
- name: Create a VirtualHost file  
  template:  
    src=vhosts.conf.j2  
    dest=/etc/nginx/sites-available/{{ vhost.servername }}  
    owner=root  
    group=root  
    mode=0644  
  notify:  
    - restart nginx
```

*module de templating*



# Ansible - Docker



## Construire une image

```
- name: build elasticsearch
  docker_image: path="/elasticsearch"
                  name="mogobiz/elasticsearch-{{es_version}}"
                  state=build
  register: build_elasticsearch
  when: elasticsearch_run|changed or elasticsearch_dockerfile|changed or elasticsearch_configuration|changed
```

*emplacement des ressources*

*nom de l'image*

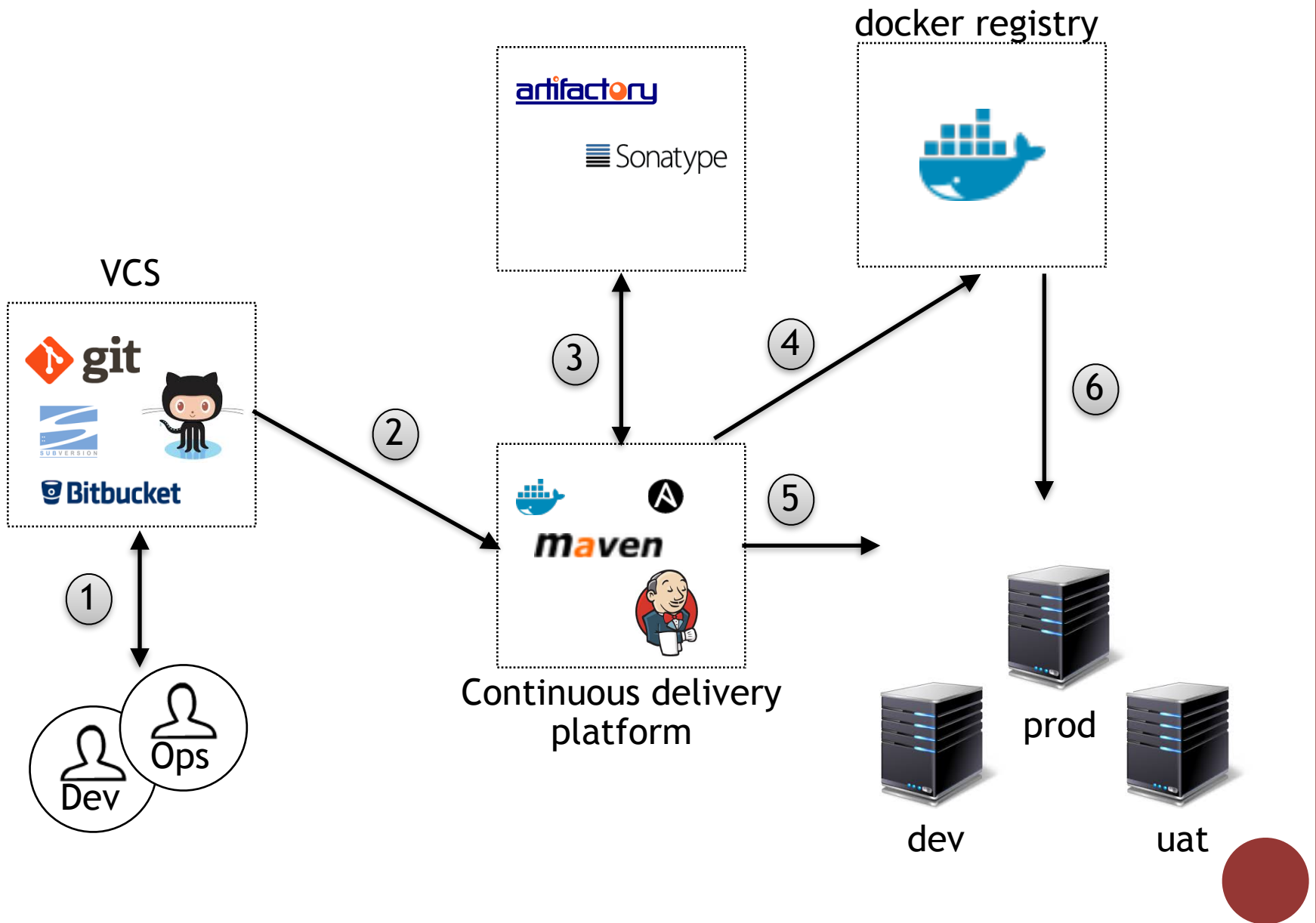
## Lancer un conteneur

```
- name: start elasticsearch container
  docker:
    image=mogobiz/elasticsearch-{{es_version}}:latest
    memory_limit=1024MB
    name=elasticsearch
    ports=9200,9300
    state=running
    volumes=/var/lib/elasticsearch:/var/lib/elasticsearch:rw,/var/log/elasticsearch:/var/log/elasticsearch:rw
  register: start_elasticsearch
  when: "elasticsearch_running is not defined or not elasticsearch_running or stop_elasticsearch_container|changed"
```

*nom de l'image + version*

*valorisation des points de montage*





# Ansible - Vagrant



```
config.vm.define "ccmu" do |ccmu|
  ccmu.vm.box = "mogobiz/precise64-docker-apache2"
  ccmu.vm.box_url = "precise64-docker-apache2.box"
  ccmu.vm.hostname = "vagrant-ccmu.vm"
  ccmu.vm.network "private_network", ip: "192.168.56.200"
  ## for mesos web UI.
  #ccmu.vm.network :forwarded_port, guest: 5050, host: 5050
  ## for Marathon Web UI
  #ccmu.vm.network :forwarded_port, guest: 8080, host: 8080
  # for oracle
  ccmu.vm.network :forwarded_port, guest: 1521, host: 1521
  ccmu.vm.synced_folder "./", "/vagrant", disabled:true
  ccmu.vm.provider "virtualbox" do |vb|
    vb.name = "vagrant-ccmu"
    vb.cpus = 4
    vb.memory = 8*1024
  end
end
```

*box*

*vm network*

*vm provider*

```
config.vm.provision :ansible do |ansible|
  ansible.inventory_path = "vagrant-inventory.ini"
  ansible.playbook = "ccmu.yml"
  ansible.extra_vars = { user: "vagrant" }
  ansible.sudo = true
  ansible.limit = 'all'
end
```

*provisioning*

*playbook*

*vagrant inventory*

# Ansible - Vagrant



```
#
# Inventory for provisioning with Vagrant
#

#####
# Local Environment #
#####
192.168.56.200 ansible_ssh_user=vagrant ansible_ssh_pass=vagrant ssh_port=2222

[mesos]
#192.168.56.200

[docker]
192.168.56.200

[oracle]
192.168.56.200

[apache]

[nginx]
192.168.56.200

[docker:vars]
oracle_hostname="{{groups['oracle'][0]}}"
```



Questions ?

Plus d'informations sur [blog.ebiznext.com](http://blog.ebiznext.com) et  
[github.com/ebiznext](https://github.com/ebiznext)

