

Infrastructure as Code with test approach

Enrique Carbonell

DevOpsDays Cuba
October 2016

About me

- BS Computer Science
- Software Engineer
- Team Leader
- Ops team

DATYS

- **DevOps Enthusiast**

Twitter: @kikicarbonell

LinkedIn: /enrique-carbonell





Client perspective

- Emplear aplicaciones y servicios de calidad:
 - Que cubran una necesidad
 - Que sean fáciles de usar
 - Que respondan en el tiempo esperado (mínimo)
 - Que siempre estén disponibles
 - Que sean seguras
 - Que....

Bussiness provider perspective

- Speed
- Innovation
- Availability
- Stability

Speed

- Time to market matters
- Cycle time matters

Innovation

- Collaboration
- Experimentation
- Rapid validation
- Market needs

Availability

- Downtime



Stability

- Infrastructure trust
- Mean time to recovery (MTTR)
- Mean Time Between Failure (MTBF)

IT Considerations?

- Hosting
- Dependencies
- Installation
- Configuration
- Updates

90's style

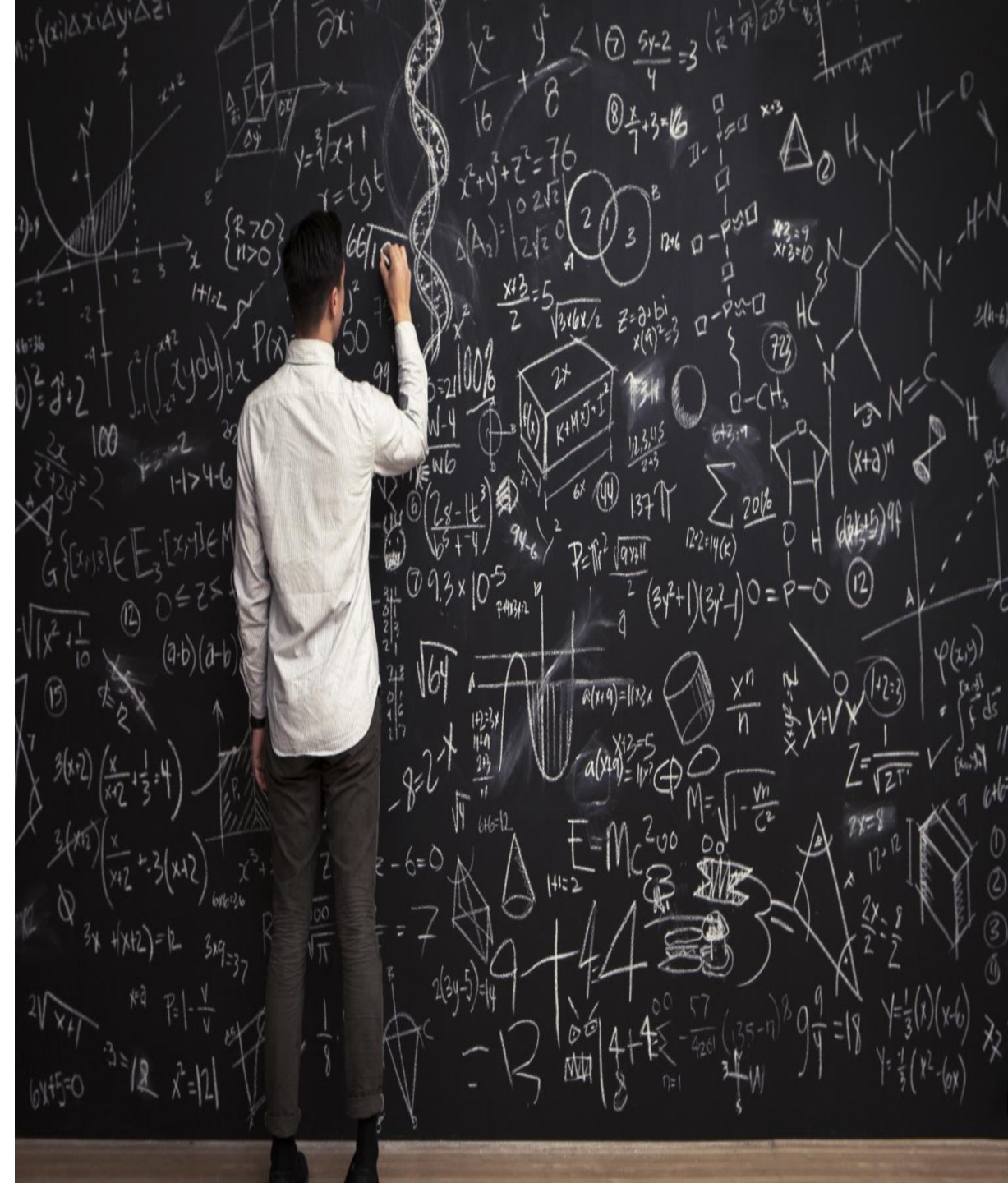
- **Manual** installation
- **Manual** configuration
- **Manual** updates
- **Manual** fixtures
- **Manual** scale
- **Manual** ...





Consequences

higher complexity



Consequences

Know-How of SAMURAI



Consequences

Concentration of knowledge



Consequences

No documentation or poor
documentations of procedures



Consequences

High probabilities of failure







SOLUTIONS?

Configuration Management Tools?

“The configuration management tools (CMT) essentially allow automation of installation, configuration and update of system’s software.....”

“Comparison of configuration management tools”

Informática 2016

Configuration Management Tools



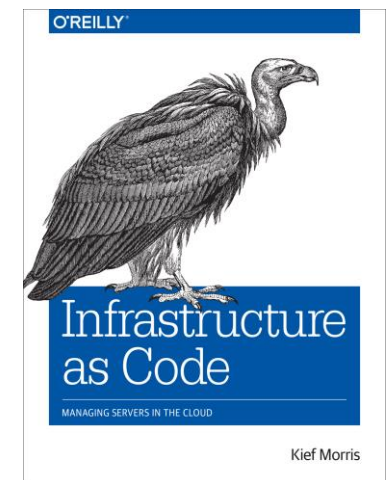
Infrastructure as Code

Infrastructure as Code (IaC)

“...is an approach to infrastructure automation based on practices from software development. It emphasizes **consistent, repeatable routines for provisioning and changing systems and their configuration...**”

Kief Morris

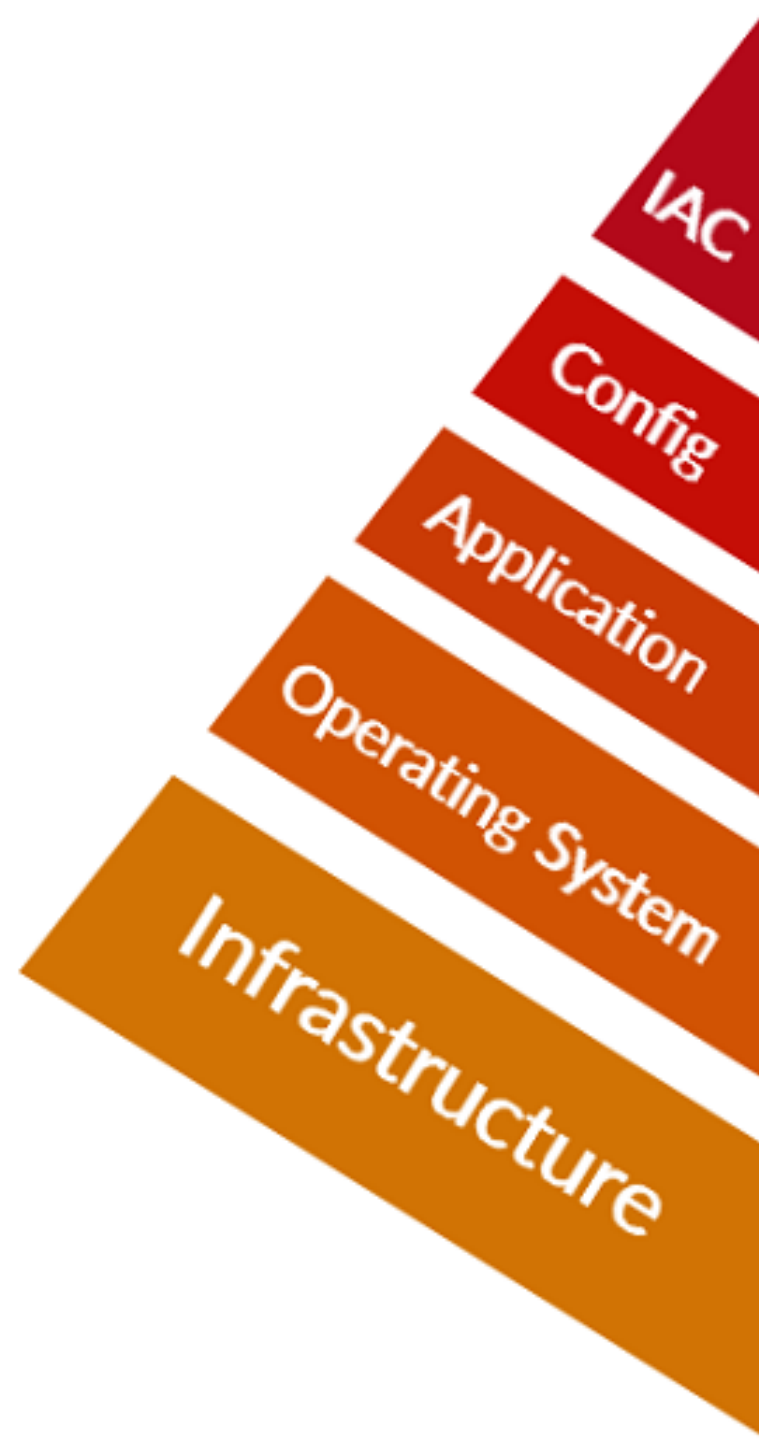
Book: Infrastructure as Code 2016



Infrastructure as Code (IaC)

“...is the process of **managing** and **provisioning** computing infrastructure (processes, bare-metal servers, virtual servers, etc.) and their **configuration** through machine-processable **definition files**, rather than **physical hardware configuration or the use of interactive configuration tools**. The definition files may be in a version control system. This has been achieved previously through either scripts or **declarative definitions**, rather than manual processes...”

Wikipedia 2016



Who is using them?



Goals of IaC

- Changes to the system are routine, **without drama or stress for users or IT staff.**



Goals of IaC

- **IT staff spends their time on valuable things that engage their abilities, not on routine, repetitive tasks.**
- Users are able to define, provision, and manage the resources they need, without needing IT staff to do it for them.

Goals of IaC

- Teams are able to easily and quickly recover from failures, rather than assuming failure can be completely prevented.
- Improvements are made continuously, rather than done through expensive and risky “big bang” projects.
- **Solutions to problems are proven through implementing, testing, and measuring them, rather than by discussing them in meetings and documents.**

Goals of IaC

- Work as developer
- Code is documented
- Peer review and pairing
- Continuous delivery
- Auditing

Work as developer

- Code is portable
- Code is reusable
- Code is version controlled

Peer review and pairing

- Code can be developed by anyone
- Code changes can be reviewed by anyone

Continuous delivery

- Code is repeatable
- Code is shareable
- Code is promotable
- Code is testable

Auditing

- Code execution can provide detailed, accurate report on actions taken
- Code promotion and execution provide authorization records
- Code run reports provide defensible audit trails

Test Driven Infrastructure (TDI)

How do it?

- Syntax check
- TDD
- BDD



Tools and resources

KitchenCI



rspec-puppet

RSpec tests for your Puppet manifests

nylas / **ansible-test**

Ansible



ANSIBLE



Why Ansible in Cuba?

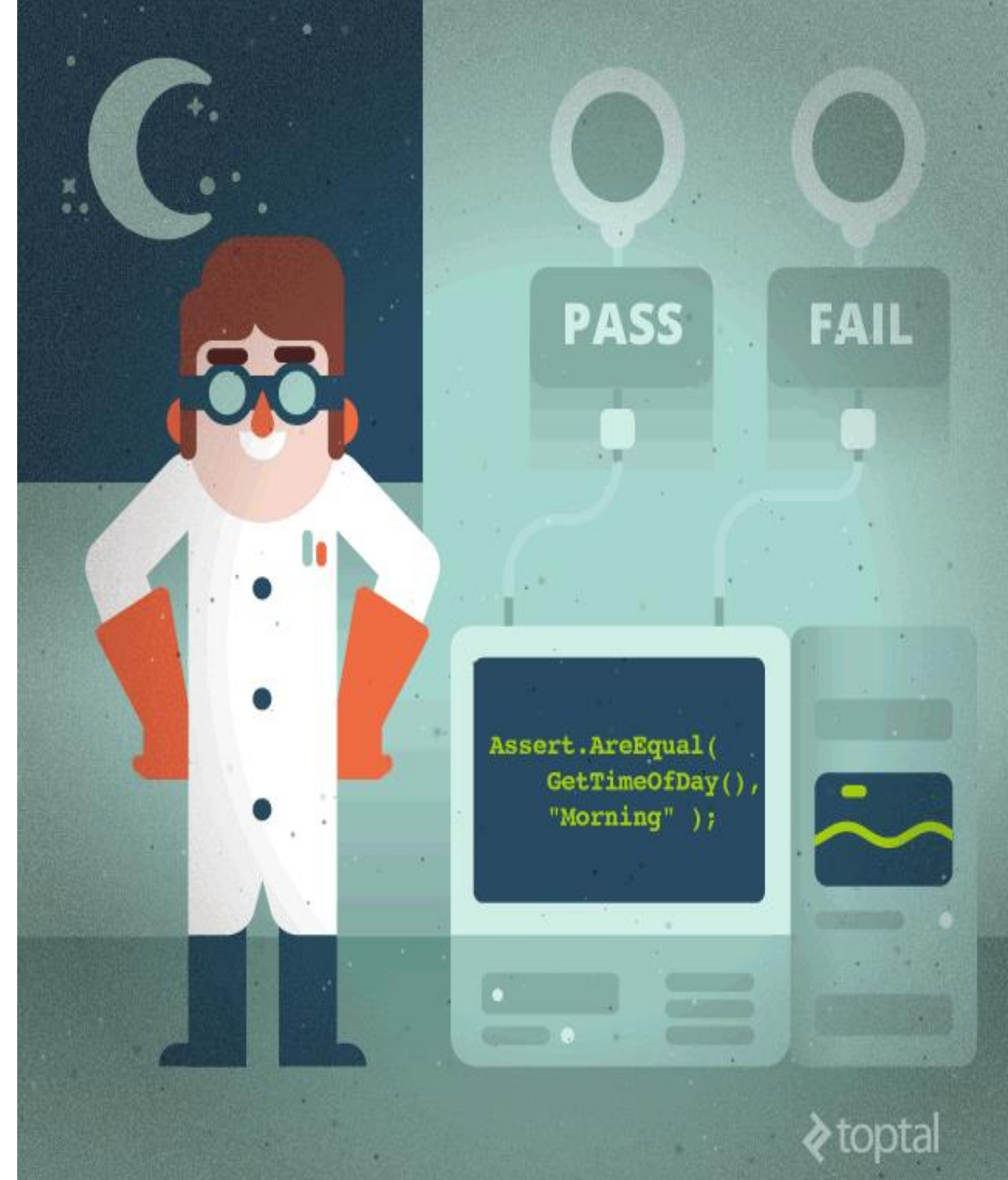
Test: Use new tools Vs known tools
















Foto via Shutterstock






Our role test






- Syntax check
- First success
- Idempotence check
- Unit test
- Behavior test



Example: nginx role structure

Ansible roles > ansible-nginx >	
Name	
 .git	
 defaults	
 handlers	
 meta	
 tasks	
 templates	
 tests	
 vars	
 .gitignore	
 .gitlab-ci.yml	
 example-vars.yml	
 README.md	
 requirements.yml	

Ansible roles > ansible-nginx > tests	
Name	
 CentOS	
 commons	
 OracleLinux	
 Ubuntu	
 .test-suite	

Ansible roles > ansible-nginx > tests > CentOS	
Name	
 all.yml	
 behavior-test.yml	
 inventory	
 unit-test.yml	
 use-case.yml	

Example: nginx role use-case.yml

```
1  ---
2
3  - hosts: localhost
4
5    roles:
6      - role: nginx
7        nginx_sites:
8          default:
9            - listen 80
10             - server_name _
11             - root "/usr/share/nginx/html"
12             - index index.html
13
14
```

Example: nginx role unit test.yml

```
1  ---
2
3  - hosts: localhost
4
5    tasks:
6      #Check package installed
7      - name: Add Nginx Test | Post-Assertions | Check nginx was installed
8        shell: nginx -v
9        register: nginx_package_installed
10       ignore_errors: yes
11
12      - fail: msg="nginx package not installed."
13        when: nginx_package_installed | failed
14
15      #Check service status
16      - name: Add Nginx Test | Post-Assertions | Check nginx service
17        shell: service nginx status | grep "running"
18        register: nginx_service_status
19        ignore_errors: yes
20
21      - fail: msg="nginx service is not running."
22        when: nginx_service_status.stdout == ""
```

Example: nginx role behavior-test.yml

```
1  ---
2
3  - hosts: localhost
4
5    tasks:
6      # Check provision expected resources
7      - name: Add Nginx Test | Post-Assertions | Check nginx provision resources
8        shell: curl http://localhost:80 | grep "Welcome to"
9        register: nginx_web_content_status
10       ignore_errors: yes
11
12      - fail: msg="index page was not provided by nginx."
13        when: nginx_web_content_status.stdout == ""
```

Our playbook test

- Syntax check
- First success
- Idempotence check
- Integration test



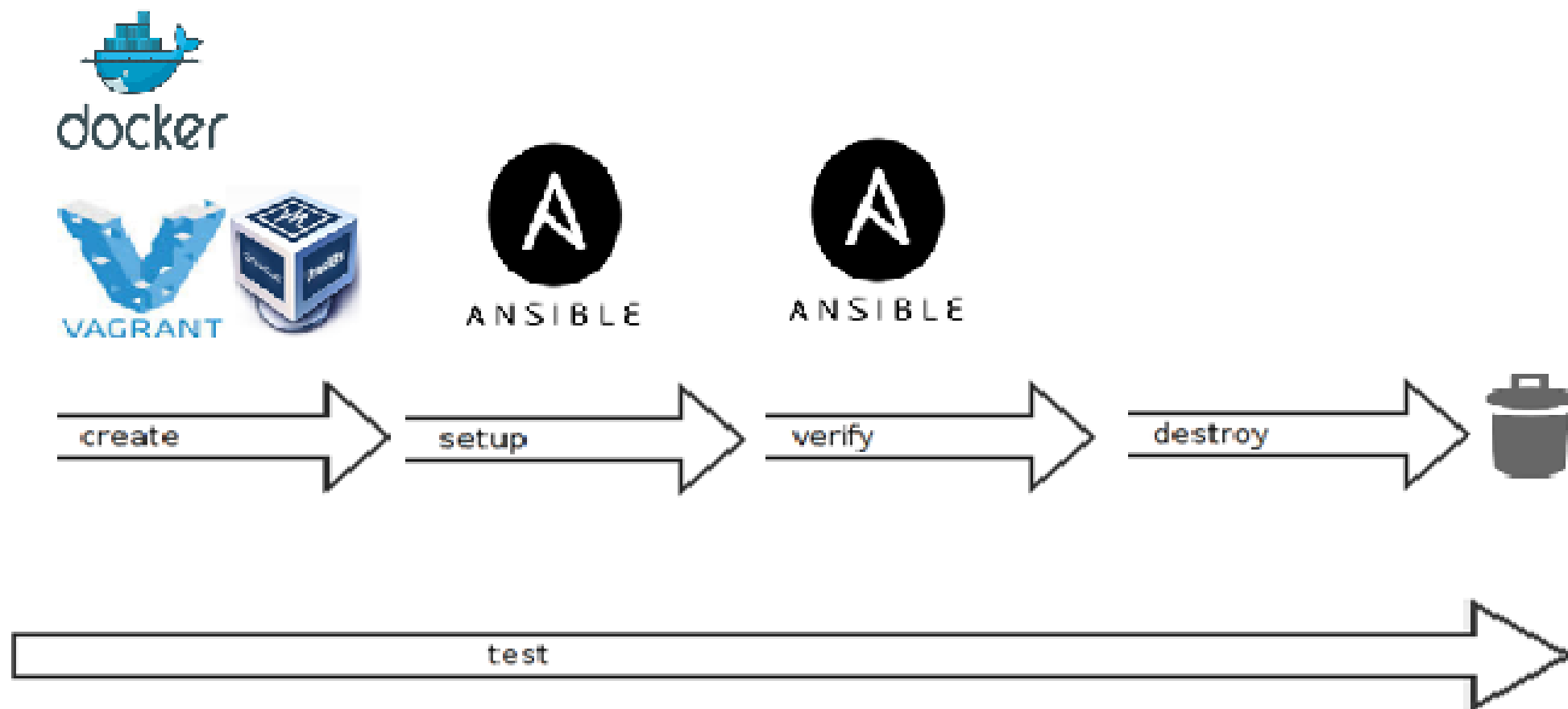
laC test environment



+



Test pipeline



Recomendation

- Have a sysadmins as coworker of developers
- Automate the most repetitive tasks in Ops
- **Test...Test...Test**