Test-driven development na infraestrutura

“Todo código é culpado até que se prove sua inocência.”

- ✔ Test-driven development
- ✔ Chef
- ✔ Docker
- ✔ Kitchen CI
- ✔ Inspec

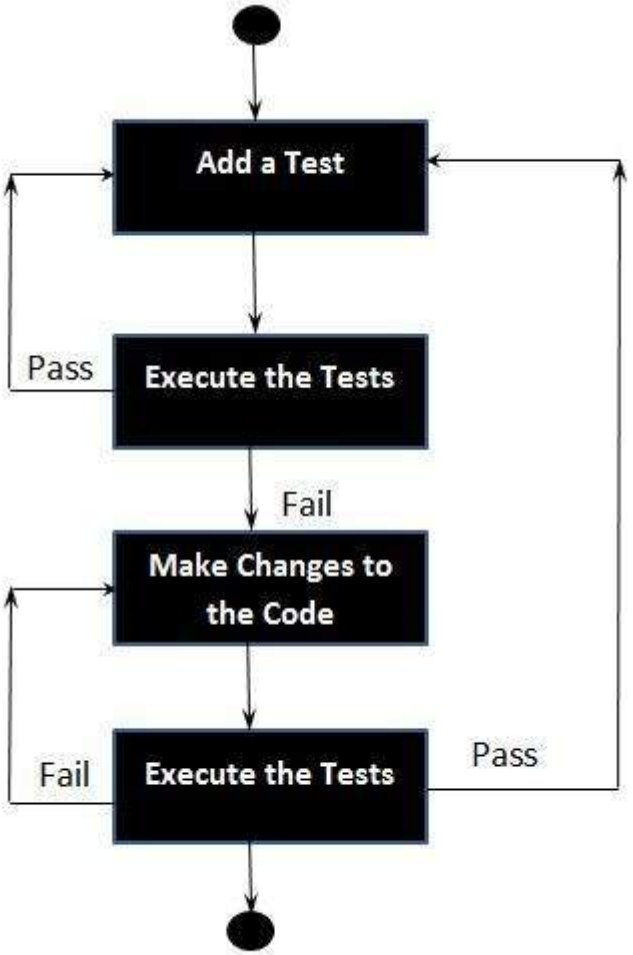
- Gerenciador de configurações
- Provisionador
- Escreva toda sua infra como código (Infrastructure as a code)
- Bare-metal, cloud, services, etc.

HashiCorp

Vagrant

- Máquinas virtuais portáteis
- Qualquer sistema operacional
- Virtualbox, Libvirt, LXC, Hyper-V
- Azure, EC2/AWS, Digital Ocean
- Chef, Ansible, Salt, Puppet
- Docker

- Originalmente desenvolvido para Chef
- Interface de execução de testes
- Inúmeros test-frameworks: Inspec, Pester, Bats
- Chef, Salt, Ansible, Puppet
- Tecnologias de virtualização: Vagrant, AzureRM, EC2/AWS, Docker, Rackspace



Pass. Development Stops

Passos do test-driven development

I. Adiciona/escreve um ou mais testes.

II. Execute os testes e tenha certeza de que falharão.

III. Escreva algum código para que um ou mais testes passem.

IV. Execute os testes para garantir que um ou mais passem.

V. Repita I a IV até que todos os testes passem.

VI. Refactore o código.

Vantagens

- ✔ Na infra se torna mais fácil de adotar TDD. Impulsona a usar o método no desenvolvimento de aplicações.
- ✔ Foco no objetivo do código.
- ✔ Minimiza tempo de debug. Você sabe com mais facilidade qual determinado bloco falhou.
- ✔ Ajuda a documentar.
- ✔ Tratamos tanto o código quanto o teste como igualmente importantes.

- ✔ Servir página html contendo ‘Hello World’
  - ✔ Container image Nginx
  - ✔ VM host com docker

- ChefDK (Kitchen CI, Inspec)
- Virtualbox
- Vagrant

OBRIGADA

Aline Freitas
aline@alinefreitas.com.br
https://github.com/alinefr
https://fb.com/aline.freitas1