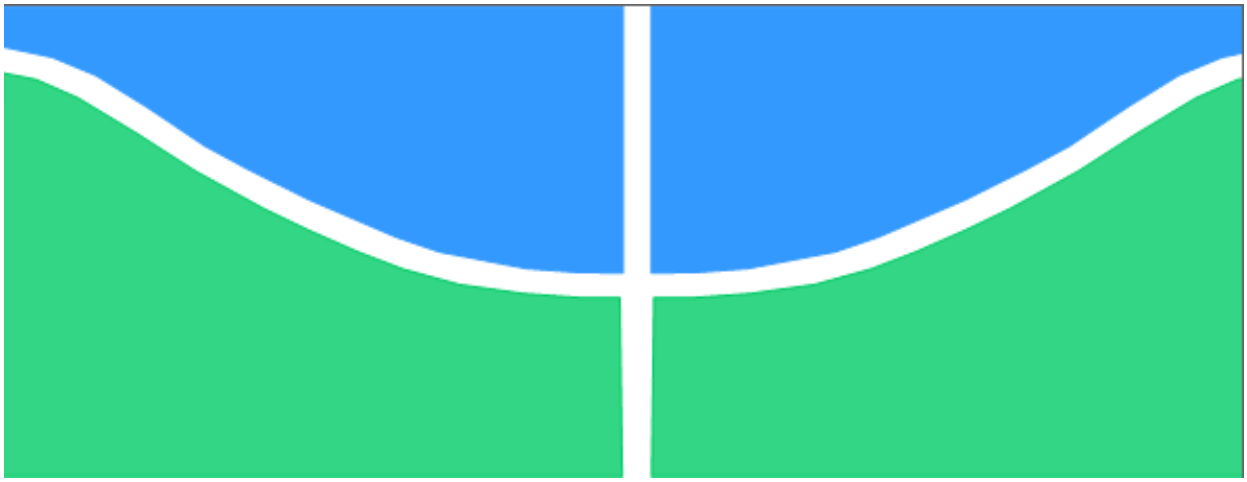

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA

Gerência de Configuração de Software

Autores:

Carolina Ramalho
Fagner Rodrigues
Matheus Fernandes

Brasília, DF
2014

Sumário

1) Radar Parlamentar

O Radar Parlamentar é um aplicativo que ilustra as semelhanças entre partidos políticos com base na análise matemática dos dados de votações que ocorrem na casa legislativa. As semelhanças são apresentadas em um gráfico bidimensional, em que círculos representam partidos ou parlamentares, e a distância entre esses círculos representa o quão parecido os mesmos votam.

Recentemente a equipe de desenvolvedores do Radar Parlamentar teve um problema no servidor da aplicação tournou-se necessária a migração para um outro servidor. Porém, infelizmente, esse processo não é tão trivial e demanda muito tempo já que a operação de deploy do Radar não é automatizada. Este, portanto, será um dos focos do projeto.

2) Tecnologias Utilizadas

- **Fabric:** é uma lib Python, e ferramenta de linha de comando, para auxiliar na parte de deploy e na administração remota dos seus sistemas. Embora seja feita em Python, pode ser utilizada em projetos escritos em qualquer linguagem. Equivale ao Capistrano, alternativa mais famosa, escrita em Ruby.
- **Chef:** é uma ferramenta de automação de infra-estrutura e utiliza uma DSL Ruby para criar os scripts de ambiente. Possui suporte a Windows, mas é mais utilizado em sistemas Linux. O Chef permite com que a build, o deploy e a gerencia de infra-estrutura sejam automatizados, versionados e testados. Utilizando as Receitas, que são conjuntos de instruções de configuração (para configurar banco de dados, por exemplo), o Chef consegue descrever o que consiste a infra-estrutura do sistema e como cada parte dela deve ser implantada, configurada e gerenciada.

- **Vagrant:** é uma ferramenta que ajuda na criação da infraestrutura para o projeto, usando para isso uma máquina virtual. A grande jogada é que o Vagrant deixa muita coisa invísivel, deixando com que você se preocupe apenas com seu código. É uma máquina virtual reduzida e portável facilmente.
- **Jenkins:** é uma ferramenta open-source de integração contínua escrita em Java, que suporta vários tipos de controles de versão, e dentre eles o Git.
- **SonarQube:** é uma ferramenta Open Source de inspeção de código. Com ele é possível extrair várias métricas que dizem respeito à qualidade do código, de diversas linguagens, além de manter um histórico de métricas.

3) Resultados

Toda a aplicação do radar estava em único repositório, o que fazia com que o repositório fosse baixado duas vezes, uma vez para ter acesso às receitas e novamente pela própria receita. Sendo assim, o repositório foi dividido em dois: o radar, propriamente dito, e um repositório de implantação, que contêm o ferramental para criar e configurar todo o ambiente utilizado para trabalhar com o radar.

Feito isso, a receita do radar, que já tinha sido iniciada pelo Leonardo, foi complementada com as seguintes funções:

- Criação de um usuário no Django, com privilégios de administrador, para ter acesso à url de importação dos dados;
- Criação de script para efetuar login no sistema, gerenciar cookies e acessar urls de importação assíncrona dos dados do radar, tudo via curl.

O próximo passo foi, então, subir e configurar o Jenkins. Para instalar o Jenkins foi utilizada uma receita encontrada no Chef SuperMarket e para a configurar foi criada uma receita dentro do cookbook “radar” chamada “jenkins.rb”. Resumidamente, tal receita tem as seguintes funções:

- Download do plugin do Git, para que possamos adicionar o repositório do radar;
- Criação dois jobs: um de build e um de deploy, através de templates do Chef;
- Instalação do Fabric.

O job de build adiciona o repositório do radar, instala as dependências utilizadas no projeto e roda os testes. Caso o job de build seja executado com sucesso este chama o job de deploy. Tal job tem como responsabilidade executar o Chef, para que o mesmo rode as receitas e faça o deploy da aplicação. Para executar o Chef utilizamos o Fabric, pois, dessa forma, conseguimos ter acesso à comandos como sudo, que é necessário para poder executar as receitas do Chef corretamente.

Feito isso, foi instalado e configurado o Sonar, para fazer uma análise estática do código. Para instalar foi utilizado o cookbook Sonarqube, e para configurar foi feita uma receita chamada “sonar.rb” que pode ser encontrada no cookbook radar. A receita de configuração do Sonar consta das seguintes funções:

- Instalação da openjdk, dependencia do Sonar;
- Criação de usuario e banco de dados usados pelo Sonar;
- Instalação do SonarRunner;
- Criação dos arquivos de configuração do Sonar e SonarRunner, através de templates do Chef;
- Execução do SonarRunner

O Sonar foi, então, incorporado ao job do Jenkins, para que ele pudesse ser executado depois que todos os testes fossem rodados.

4) Dificuldades Encontradas

a. Travis CI

Inicialmente, ao elaborar o Plano de Gerência de Configuração, tínhamos sugerido de substituir o Travis CI pelo Jenkins, por ter visto mais conteúdo sobre a integração dele com o Fabric, que seria nossa sugestão para realizar o deploy. Porém, ficou decidido que o Travis seria mantido, pois já estava suprindo as necessidades e seria um trabalho, talvez, desnecessário.

Ao finalizar o desenvolvimento da receita, começamos a pesquisar formas de realizar o deploy automático, a cada commit. Após esses estudos vimos que para realizar o deploy automático, seria necessário deixar especificada a senha do servidor, para que o Travis conseguisse executar comandos remotamente. Isso inviabilizou o uso do Travis como ferramenta de integração contínua.

A solução encontrada, foi a sugerida no inicio do projeto: utilizar o Jenkins para a integração contínua. Dessa forma, o Jenkins poderia acessar arquivos da própria máquina, até mesmo executar comandos localmente, o que torna o deploy mais fácil de ser realizado. Porém, ao configurar o job de deploy da aplicação, muitos erros de permissão ocorreram e para resolvê-los, passamos a utilizar o Fabric para executar comandos usando ‘sudo’ na própria máquina. Dessa forma, o fluxo final da integração ficou:

1. A cada 5 minutos o Jenkins verifica se houve alteração no repositório.
2. Se houver alteração:
    a. Executa a suite de testes
    b. Executa a análise estática, usando o Sonar-Runner
    c. Se todos os testes passarem com sucesso:
        i. A build é gerada
        ii. O Jenkins executa o job de deploy

iii. O job de deploy chama o Fabric, que por sua vez, executa o Chef na própria máquina

iv. O deploy é executado e a alteração realizada pelo commit é colocada em produção.

b. Diferença entre versões do Chef

Inicialmente, a receita do Chef para o deploy do radar foi escrita utilizando a versão 10 do Chef, o que impossibilitava a utilização de alguns cookbooks do SuperMarket, pois tinham como pré-requisito a versão 11 do Chef. O uso desses cookbooks facilitariam bastante o trabalho de escrever as receitas de instalação do Sonar e do Jenkins, portanto, a migração da receita para a versão 11 do Chef foi realizada.

O script que estava sendo utilizado para instalar o Chef-Solo sempre utilizava a versão mais recente do Chef, o que resultou em erros durante os últimos testes, pois foi lançada a versão 12 e a receita do radar não estava de acordo com os padrões da nova versão. Para evitar esse tipo de conflito os script de instalação foi atualizado, passando a utilizar o gerenciador de pacotes do Ruby, o gem.

c. Configuração de jobs no Jenkins

A maior dificuldade para a configuração dos jobs no Jenkins foi o erro de “Permission Denied” quando o Jenkins tentava rodar qualquer comando que necessitasse utilizar o sudo. Foram utilizadas várias alternativas para contornar o problema, como por exemplo adicionar um usuário com acesso total no arquivo sudoers, porém não funcionou. O problema foi contornado então utilizando o Fabric para executar comandos como sudo.

d. Integração Vagrant + Chef 11

Para que o ambiente dos colaboradores do projeto seja o mais fiel possível ao de produção, evitando, assim, possiveis erros que só acontecem em ambiente de produção, utilizaremos o Vagrant. A box que até então vinha sendo utilizada em conjunto com o Vagrant, ja tinha o Chef integrado e, ao provisionar a máquina a receita do Chef era executada, realizando o deploy da aplicação. Porém, como realizamos a migração para a versão 11 do Chef essa box, com o Chef v10, não serviria mais. Não foi fácil, mas conseguimos encontrar uma box já com a versão mais recente e a partir daí tudo funcionou perfeitamente.

Portanto, hoje é possível desenvolver para o Radar Parlamentar utilizando um ambiente bem parecido com o de produção em que podemos prever e evitar erros em produção que não poderiam ser previstos em ambientes de desenvolvimento, como já aconteceu antes.