README.md

Instalando o Jenkins como ferramenta de C.I.

Objetivo: Instalar o Jenkins e configurar um webhook para deploy de código dentro da plataforma criada em node;

Afinal o que é o Jenkins?

O Jenkins é uma aplicação opensource baseado em java utilizado no processo de automação de tarefas, essas tarefas vão desde o processo de building de código até testes automatizados e deploys em ambientes de produção, sua instalação necessita apenas de ambiente com Java e também pode ser executada na mairoai dos sitemas operacionais em uso atualmente ou mesmo a partir de containers enquanto a arquitetura pode ser distribuida ou no formato standalone com apenas uma VM.

> Em nosso cenário o Jenkins foi a solução escolhida como ferramenta para implementação de Deploy Continuo e parte da arquitetura para integração continua do código no ambiente criado no Laboratório anterior.

O processo de instalação do Jenkins no formato standalone, ou seja, utilizando apenas um servidor é relativamente simples, basta seguir a documentação oficial no site do Projeto atráves da URL jenkins.io/doc, em nosso cenário a instalação será feita no Ubuntu Server 16.04 essa instalação será baseada em outra Doc específica, a Wiki do Projeto Jenkins;

Executando a instalação do Jenkins

1. Para executar a instalação em distribuições baseadas em Debian, como nosso recém configurado ubuntu server, você pode instalar o Jenkins através da ferramenta apt-get que é um gerenciador de pacotes, o processo de instalação usando o apt é bem simples uma vez que está bem documentado, basta seguir conforme abaixo:

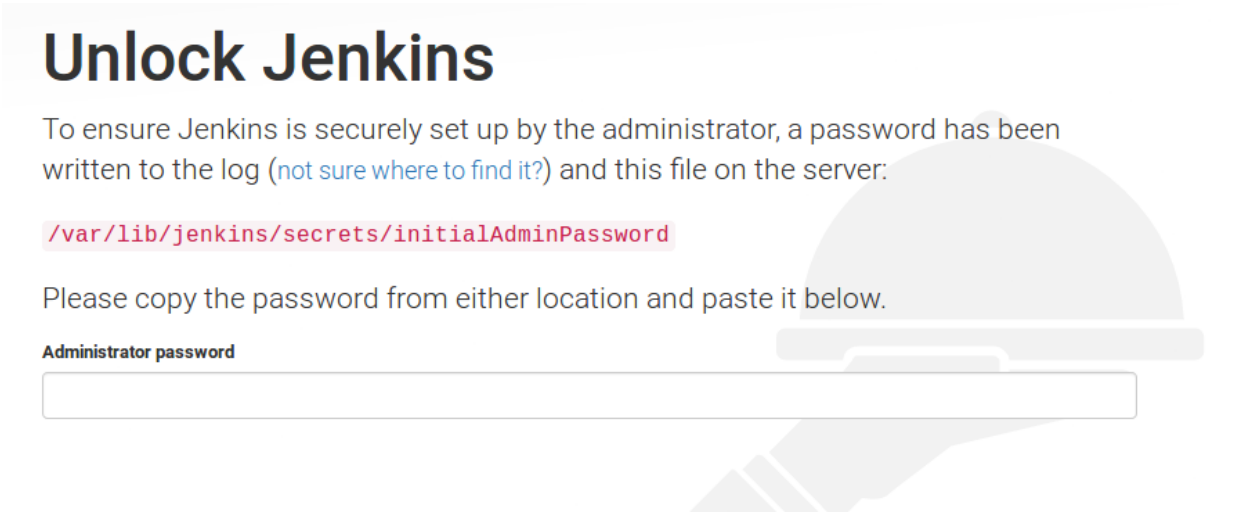
```
# wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
# sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
# sudo apt-get update
# sudo apt-get install jenkins
```

> A instalação do Jenkins exige que uma VM JDK e JRE esteja instalada, em nosso caso o ambiente veio com o Java 8 do pacote openjdk-8 instalado o que resolve essa dependecia;

2. Finalizando a instalação basta inicializar a aplicação e garantir sua reinicialização automatica, para isso execute:

```
# systemctl start jenkins
# systemctl enable jenkins
```

3. A aplicação será incializada utilizando a porta ***8080*** do servidor, dessa forma abra um navegador Web e vá para http://:8080 onde concluiremos a configuração inicial do Jenkins;

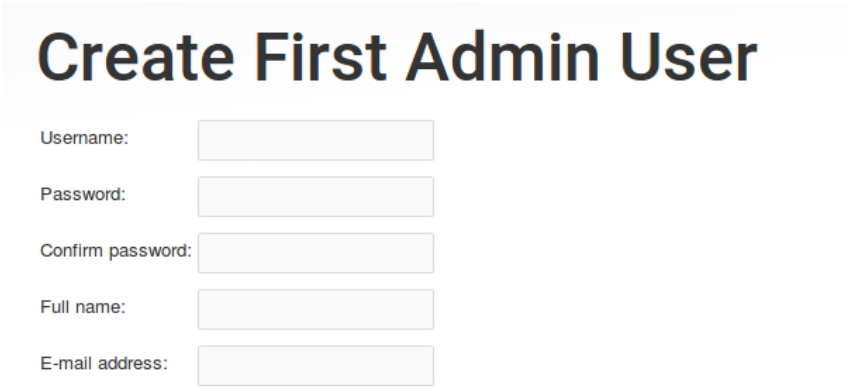


A tela acima exige que para fins de segurança seja inserido a senha de administrador inicial, esse dado está armazenado em um arquivo de texto na sua VM. Use o endereço IP público obtido na etapa anterior para conectar-se à sua VM por SSH e recolher essa informação;

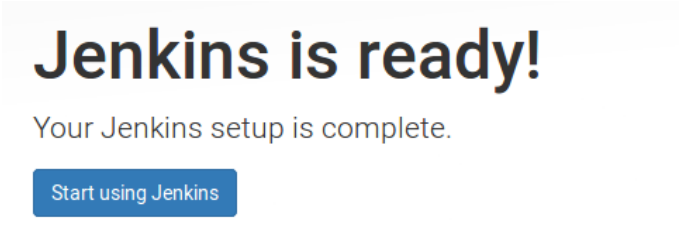
```
# sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

4. Adicione a chave de segurança e tecle enter, é possível customizar o processo de instalação do Jenkins utilizando plugins de acordo com o projeto de delivery e com a linguagem de programação, ferramentas e repositórios envolvidos, em nosso cenário utilizaremos a opção ***"Installed Suggested Plugins"***.

5. Por questões de segurança um opicional importante é criar um usuário dentro do Jenkins em vez de continuar usando a conta de administrador, para criar esta conta de usuário, preencha o formulário conforme desejado;



6. Quando terminar, clique em Começar a usar o Jenkins;

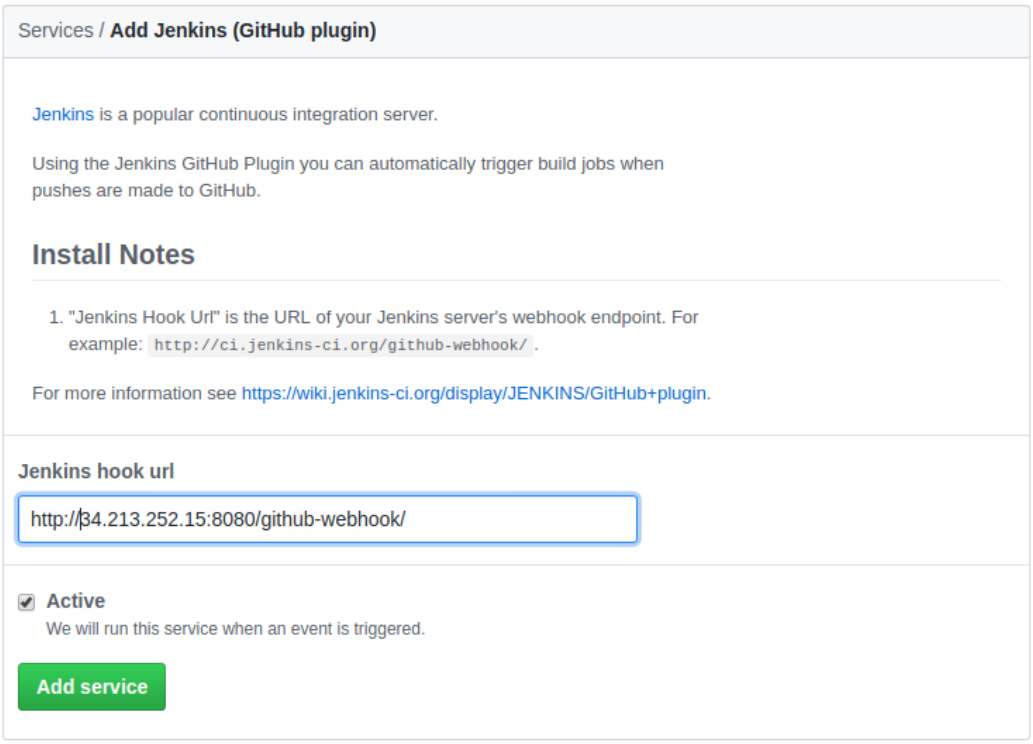


Criando um webhook no GitHub;

Para configurar a integração com o GitHub, abra o app em node que bifurcamos anteriormente (criação do fork no Lab 2.1), crie um webhook dentro da bifurcação criada:

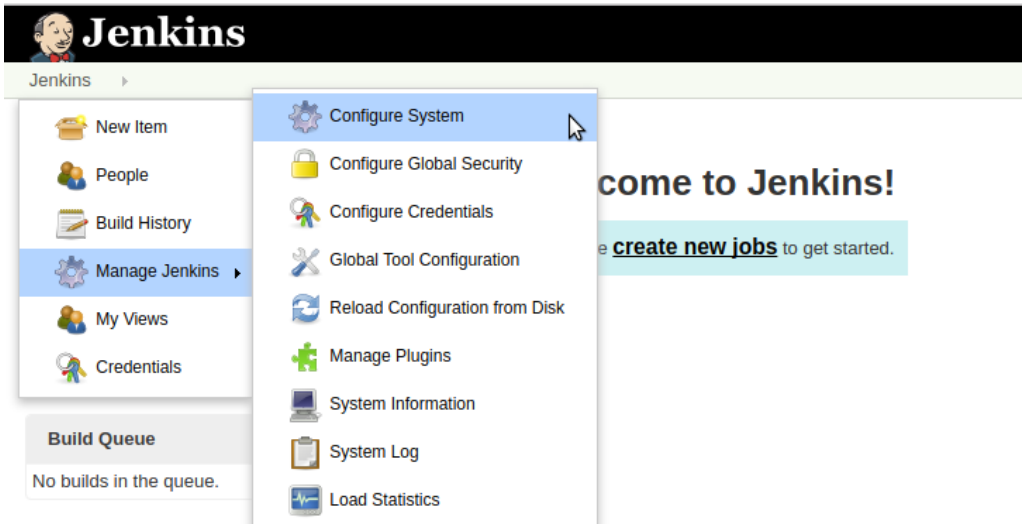
1. Clique em ***Settings*** e, em seguida, selecione ***Integrations & services*** no lado esquerdo.

2. Clique na opção ***Add service*** em seguida, digite Jenkins na caixa de filtro;
3. Selecione ***Jenkins (GitHub plugin)***;
4. Para a o campo "Jenkins Hook Url" digite http://ci.jenkins-ci.org:8080/github-webhook/. Certifique-se de incluir a barra à direita (/) e trocar o endereço ci.jenkins-ci.org pelo seu endereço ip ou DNS do servidor;
5. Clique em Adicionar serviço conforme a imagem abaixo;

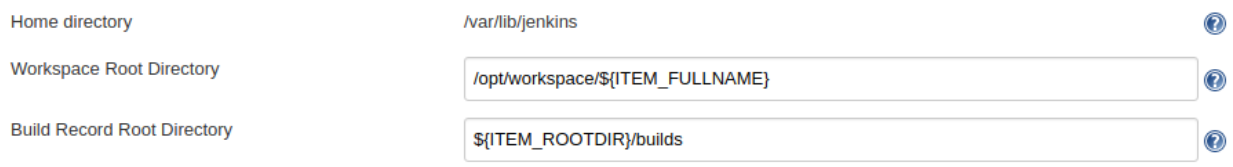


Configurando o Jenkins:

Em nosso modelo de deploy o Jenkins será responsável por manipular a pasta do Projeto, colocando o código em produção, para isso altere o workspace do Jenkins através do Menu ***Jenkins***, ***Manage Jenkins***, ***Configure System***:



Clique no botão ***Advanced...*** e altere o valor do campo ***Workspace Root Directory*** para ***/opt/workspace/${ITEM_FULLNAME}*** conforme abaixo:



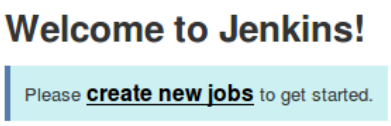
Clique em ***Save*** no canto inferior da tela para salvar sua alteração;

> Esse modelo não é ideal e deverá ser alterado em aulas futuras quando passarmos a utilizar containers mas de início é a solução mais didatica e com menor intervenção manual no sistema operacional, já que este não é o nosso foco.

Criando um Job no Jenkins;

Para que o Jenkins responda a um evento disparado no GitHub, é necessário que criemos um Job com instruções específicas sobre isso.

1. Volte ao painel em seu Jenkis e clique em ***Create new Jobs*** a partir da home page:



2. O nome do item a ser criaod será ***node-app***, insira o nome e selecione a opção ***Freestyle project*** e clique em OK;
3. Na seção ***General***, selecione a opcão ***Github project***, insira a URL do repositório, por exemplo, https://github.com/helcorin/nodejs-docs-hello-world
4. Avance para a sessão ***Source Code Management***, selecione Git e insira a URL .git do repositório, por exemplo, https://github.com/helcorin/nodejs-docs-hello-world.git, como trata-se de um repositório aberto não será necessário adicionar credenciais.
5. Na seção ***Build Triggers***, selecione a opção ***GitHub hook trigger for GITScm polling***.
6. Na seção ***Build Environment***, selecione a opção ***Delete workspace before build starts***.
7. Na seção ***Build***, clique em ***Add build step***. Selecione ***Execute shell***, em seguida, digite os comandos abaixo na janela de comando.

7. Na parte inferior da janela de trabalhos, clique em ***Salvar***.

Preparando o terreno na VM

Como até o ultimo laboratório executávamos o node com uma cópia local será necessário duas mudanças na VM para executarmos o Build e presenciarmos a alteração em produção:

1. Adicione uma permissão de execução diretamente na VM que hospeda a aplicação:

```
# visudo -f /etc/sudoers.d/91-jenkins-restart
```

Adicione o conteudo exatamente como no modelo abaixo e salve o arquivo criado

```
# Allow jenkins user to restart node-app deamon
jenkins ALL=NOPASSWD: /bin/systemctl restart node-app.service
```

2. Altere as permissões do diretório "/opt/workspace" para que ele passe a ser manipulado pelo Jenkins

```
# chown -R jenkins: /opt/workspace/
```

Testando a integração com o GitHub;

Para testar a integração do GitHub com o Jenkins, precisaremos criar um evento, isto é executar uma alteração na nosso Fork do código.

1. Para executar esse processo volte à interface Web do GitHub, selecione o repositório e em seguida, localize o arquivo ***index.js***

2. Clique no ícone de lápis para editar o arquivo, faça qualquer alteração no arquivo, como exemplo sugiro editar a mesnagem na linha 6:

3. Para confirmar suas alterações, clique no botão ***Commit Changes*** na parte inferior da interface;

4. No canto inferior esquerdo da interface do Jenkins você verá um novo processo de Build que fora iniciado conforme imagem abaixo:

5. Você verá o processo de Build cirado a partir do evento iniciado por você ao executar o commit de alterações de código no Github;

6. Clicando na opção ***Console Output*** você verá a saída de nosso Build, como ainda não estamos efetivamente fazendo o Build da forma correta iremos apenas executar dois comandos;

7. O processo de Build deverá ter adicionado o código com as alterações no workspace default do Jenkins, em nosso caso o diretório "/var/lib/jenkins/workspace" do servidor;

Importante:

RTA - Recurso Técnico Avançado (Ou Gambiarra mesmo)

Para que nosso Laboratório permita a entrega de conteúdo em produção e assim a visualização do processo completo, saindo do commit e acessando a página com a alteração algumas alterações foram necessárias, essas alterações não são definitivas e devem ser substituidas por decisões de arquitetura melhor elaboradas em um futuro próximo;

1. Em nosso modelo que ainda está BEM (mas bem mesmo) BÁSICO a estratégia escolhida foi criar uma unidade no systemd cuja função é iniciar e reiniciar a aplicação em node, esse service pode ser conferido no arquivo /lib/systemd/system/node-app.service criado no Lab anterior;

2. Para que o Jenkins pudesse executar o comando "sudo /bin/systemctl restart node-app.service" utilizei uma permissão especial de sudo entregue a partir do arquivo /etc/sudoers.d/91-jenkins-restart, na verdade nesta etapa o deploy já foi feito uma vez que o código está sendo executado diretamente a partir do Workspace do Jenkins, o comando citado é necessário apenas para reinicializar a aplicação;

Bibliografia e Referências:

O processo de instalação do Jenkins foi baseado na documentação oficial da Wiki já citada:

- Installing Jenkins on Ubuntu - https://wiki.jenkins.io;

Já a criação do webhook foi baseada em um paper publicado no site da Azure:

- Como criar uma infraestrutura de desenvolvimento em uma VM Linux no Azure com Jenkins, GitHub e Docker;

Free Software, Hell Yeah!