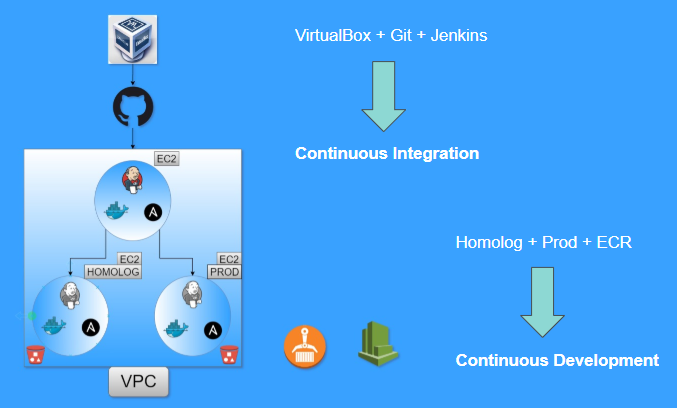
Situação: Recebimento de uma aplicação em Node.js em desenvolvimento, com funcionalidades a serem passadas por teste de qualidade.

Projeto da equipe Lovelace: Criar uma pipeline de entrega contínua da aplicação a ser armazenada na Cloud

Tecnologias utilizadas: Ansible, Docker, Jenkins, AWS

Arquitetura



Primeiros passos

1. Configurar o ambiente para executar o plano de ação:

- Instalar a VirtualBox;

- Dentro da VM, instalar os softwares VSCode, Ansible, Docker e Jenkins;

- Criar contas na AWS.

2. No desktop da máquina virtual, criar um arquivo para as senhas no VSCode ou no próprio terminal:

$ vi ~/.ansible/.vault\_pass

3. Criar o arquivo***aws\_credentials.yml*** na pasta ***vars*** do repositório do projeto com as credenciais de segurança do usuário root da AWS:

- Acessar o console da AWS com o user root e seguir o seguinte caminho: My Security Credentials > Access keys (access key ID and secret access key) > Create New Access Key;

- Colocar as o AWSAccessKeyId e o AWSSecretKey em ***aws\_credentials.yml****;*

- Copiar as informações do arquivo para futuro acesso;

- Encriptar ***aws\_credentials.yml****:*

$ ansible-vault encrypt playbooks/vars/aws\_credentials.yml

4. Criar um repositório para o projeto no Github e criar o arquivo **.gitignore**, que não permitirá que determinados arquivos sejam versionados no repositório (como ***aws\_credentials.yml****, por exemplo)*;

5. Configurar AWSCLI no desktop da VM;

6. Configurar as credenciais do usuário da AWS no desktop e colocar a região de trabalho (no momento, compensa financeiramente utilizar a *us-east-1*):

$ aws configure

$ aws iam list-access-keys

$ aws ec2 describe-regions

Execução do projeto

1. Customizar os Playbooks:

- Configurar a infraestrutura em **aws\_provisioning.yml** (arquivo principal de configurações), **aws\_provisioning\_vpc.yml**, **aws\_provisioning\_jenkins.yml**, **aws\_provisioning\_producao.yml** e **aws\_provisioning\_homolog.yml**;

- Em **aws.yml**, configurar:

* AWS\_access\_key: *"{{ AWSAccessKeyId }}"\* > Credenciais AWS* (está no arquivo criptografado **vars/aws\_credentials.yml**).
* AWS\_secret\_key:*"{{ AWSSecretKey }}"\* > Credenciais AWS* (está no arquivo criptografado **vars/aws\_credentials.yml**).
* instance\_type (tipo da instância que será criada na AWS): t2.micro;
* security\_group (nome do Security Group): "{{ projeto }}-ws-sg";
* image\_ami (imagem a ser utilizada para criar a máquina na AWS): ami-07ebfd5b3428b6f4d;
* KeyPairDev (nome da key-pair): "{{ projeto }}-key";
* region (região a ser usada na AWS): us-east-1;
* vpc\_cidr\_block: 10.0.0.0/16;
* cidr\_block: 10.0.1.16/28;
* vpc\_name: "{{ projeto }}-vpc";
* route\_tag (Nome da Route Table, que é parte da VPC): "{{ projeto }}-route";
* sub\_tag (Nome da Subnet, que é parte da VPC): "{{ projeto }}-subnet";

- Criar a Key Pair das instâncias EC2;

- Criar 3 [Amazon EC2](https://aws.amazon.com/pt/ec2/) (Amazon Elastic Compute Cloud, uma máquina virtual na nuvem) em cada playbook referente ao respectivo ambiente: **aws\_provisioning\_jenkins.yml** (a EC2 criada será o servidor Jenkins), **aws\_provisioning\_producao.yml** e **aws\_provisioning\_homolog.yml**;

- Em **aws\_provisioning\_jenkins.yml**, criar as seguintes tasks:

1. *Launch the new EC2 Instance 22* (Executar a nova instância EC2 de acordo com as variáveis do itens vars\_files e o vars);
2. *Wait for SSH to come up* (Aguardar o SSH chegar);
3. *Create AWS ECR* (Criar a *Amazon Elastic Container Registry*, um tipo de repositório que armazena e gerencia as imagens de contêineres do Docker geradas pela pipeline).

- Em **aws\_provisioning\_homolog.yml** e **aws\_provisioning\_producao.yml**, criar estas tasks:

1. *Launch the new EC2 Instance 22;*
2. *Wait for the SSH to come up;*
3. *Create iam user “{{ name\_aim\_user\_s3 }}”* (Criar o usuário IAM);
4. *Create a bucket* (Criar um bucket, recurso de armazenamento em cloud da AWS);
5. *Create S3* (Denomina o bucket criado como *Simple Storage Service*, que armazenará os dados gerados pelo ambiente - seu nome deve ser um nome único em letras minúsculas com tamanho entre 3 e 63 caracteres).

- **config\_all-ec2.yml** deve atualizar os pacotes em todas as EC2 e instalar softwares e suas dependências, como awscli, java, python;

- **install\_docker\_all-ec2.yml** deve instalar o docker nas EC2 que foram criadas;

- **install\_ansible\_ec2-jenkins.yml** deve Instalar o Ansible e pacotes para facilitar configurações;

- **install\_jenkins\_ec2-jenkins.yml** deve instalar e configurar o Jenkins na EC2 master.

2. Provisionar EC2, S3, IAM e ECR na AWS

- Criar uma VPC (*Virtual Private Cloud*) e outros itens importantes para sua utilização (como Subnet e Security Group):

- Gerar uma key pair e salvá-la;

- Criar 3 EC2 (uma para o Jenkins, uma para o ambiente de Homologação e outra para o ambiente de Produção), um ECR (*Elastic Container Registry*, que tem a função de armazenar e gerenciar as imagens de contêineres do Docker geradas pela pipeline), um *user* e um *bucket S3;*

$ ansible-playbook playbooks/aws\_provisioning.yml

- Verificar se a VPC foi criada corretamente:

$ ansible-inventory --graph aws\_ec2

3. Configurar máquinas EC2

- Atualizar pacotes e instalar AWCCli, Java e Python nas 3 EC2:

$ ansible-playbook playbooks/config\_all-ec2.yml

- Instalar o Docker nas 3 EC2:

$ ansible-playbook playbooks/install\_docker\_all-ec2.yml

- Instalar o Ansible nas 3 EC2:

$ ansible-playbook playbooks/install\_ansible\_ec2-jenkins.yml

- Instalar o Jenkins apenas na EC2 master:

$ ansible-playbook playbooks/install\_jenkins\_ec2-jenkins.yml

4. Configurar o Jenkins

- Entrar no terminal e acessar a url de acordo com o nome ou IP público gerado na AWS;

- Logar no Jenkins com usuário e senha definidos no playbook **install\_jenkins\_ec2-jenkins.yml**;

- Seguir o caminho:

*Gerenciar Jenkins > Gerenciador de Plugins > Disponíveis >* Buscar e selecionar *pipeline, docker pipeline, ssh, github, github api* e *amazon ecr;*

- Acessar a EC2 via SSH:

*ssh -i @ ssh -i .pem ubuntu@*

- Verificar o usuário Jenkins:

$ id jenkins

- Incluir o usuário Jenkins no grupo Docker do Linux:

$ sudo addgroup jenkins docker

- Parar o serviço do Jenkins:

$ sudo service jenkins stop

- Subir o serviço do Jenkins:

$ sudo service jenkins start

- Validar o serviço do Jenkins:

$ sudo service jenkins status

- Configurar nodes Homolog e Produção para a execução do pipeline:

*Gerenciar Jenkins > Gerenciar nós > novo nó*

- Adicionar Credenciais para acessar as instâncias:

*Credentials > Add Credentials* (Global, SSH username e private key)

- Adicionar Credenciais para acessar AWS ECR:

*Credentials > Add Credentials* (AWS Credentials, Global, ID e Private Key AWS)

5. Pipeline Jenkins

Com a aplicação pronta para release, é possível criar um Job, cuja função será realizar o deploy de forma automática.

Aplicação Node.JS utilizada

<https://github.com/bgsouza/digitalhouse-devops-app>

Evidências

* Pipeline
  + [Pipeline](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/PipelineOk.png) executado com sucesso:
  + [HealthCheck Homologação](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/healthcheck-homolog.png) executado com sucesso:
  + [HealthCheck Produção](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/healthcheck-prod.png) executado com sucesso:
  + [Output Pipeline](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/Output-Pipeline.pdf) de todos os passos:
* Url
  + [HealthCheck Homologação](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/url-healthcheck-homolog.png) executado com sucesso:
  + [HealthCheck Produção](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/url-healthcheck-prod.png) executado com sucesso:
  + [Upload Homologação](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/url-upload-homolog.png) executado com sucesso:
  + [Upload Produção](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/url-upload-prod.png) executado com sucesso:
* AWS S3
  + [Buckets S3](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/Buckets-S3.png) criados:
  + [Bucket S3 Homolog](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/BucketS3-homolog.png) com a imagem:
  + [Bucket S3 Produção](https://github.com/andresavs/ProjetoIntegrador_AcademiaNeon/blob/andresa/docs/BucketS3-prod.png) com a imagem:

Referências

* Professores
  + Bruno G. Souza - <https://github.com/bgsouza/digitalhouse-devops-app>
  + Krishna Pennacchioni - <https://github.com/agentelinux/devops-pi/tree/grupo1>
* Material do curso - Playground Digital House
* Documentação Oficial
  + <https://www.ansible.com/>
  + <https://galaxy.ansible.com/>
  + <https://github.com/>
  + <https://www.docker.com/>
  + <https://www.jenkins.io/>
  + <https://aws.amazon.com/pt/>
  + <https://www.markdownguide.org/>