Insper

Computação em Nuvem

Cap. 0 - Public Cloud Intro - APS

Raul Ikeda - rauligs@insper.edu.br

Nome:

Usuário Git:

Objetivos

1. Introduzir os conceitos básicos sobre cloud computing.
2. Entender os conceitos básicos sobre IaaS.

Pré-requisitos:

1. Roteiro individual.
2. Realizar leitura prévia:
   [https://aws.amazon.com/pt/products/]
   [http://redhatstackblog.redhat.com/2015/05/13/public-vs-private-amazon-compared-to-openstack/]
3. Realizar a leitura sobre Cloud Computing. [Kavis - cap. 2 e Earl et al - cap. 4]

Primeira Instância

Vamos começar conhecendo o ferramental básico da AWS para *virtual machines*.

- Acessar o dashboard do EC2 de *North Virginia* (**us-east-1**).
- Criar uma instância t2.micro com Ubuntu 16.04 LTS server e 8Gb de disco.
- Renomear a máquina com o seu nome (*Tag*).
- Verifique se o *security group* liberou a conexão SSH externa.
- Importar em *Key Pair* a sua *public key* gerada na primeira aula. Coloque o seu nome na chave.

  - Endereço IP público da máquina:

- Teste a conexão SSH (não se esqueça da chave privada).

1. Você criou uma policy para seu usuário. O que são *Policies*? Por que elas são importantes e devem ser bem definidas?

2. Dentro do contexto de Cloud Computing, defina os termos: *instance*, *image*, *region*, *VPC*, *subnet*, *security-group*.

3. O poder computacional das instâncias é medido em vCPU. O que é vCPU?

Primeiro Deploy - Ghost Blog Platform

Vamos fazer um deploy completo e rápido com poucos comandos. Acesse o terminal da instância criada e execute os seguintes comandos:

- Atualize o S.O.
  - $ sudo apt update && sudo apt upgrade -y
- Instale o conjure-up usando o snap
  - $ sudo snap install conjure-up --classic
- Agora instale a plataforma Ghost
  - $ conjure-up battlemidget/ghost
  - Selecione aws e a região correspondente (us-east-1)
  - Entre com o par *Access Key ID*/*Secret Access Key*
  - Selecione *Self-hosted controller*
  - Deploy!
  - Mantenha a URL como localhost.
- Teste o deploy
  - $ juju status
  - Anote o IP da aplicação HAProxy.
  - Acesse a plataforma: https://[IP Público]/admin

1. Quantas instâncias foram criadas automaticamente? Criar instâncias automaticamente é um atributo positivo ou negativo?

2. Quanto custou o protótipo? Assuma que usou uma semana de cada instância.

3. Dada a quantidade de computadores apontada na questão anterior, descreva como você montaria um ambiente Ghost em um Datacenter próprio. Assuma que você ainda não possui nenhum hardware disponível, apenas um orçamento aprovado.

4. Agora, somando o fato de que hardware deprecia com o tempo e possui um custo mensal de manutenção, compare em termos de custo uma *Public Cloud* e um Datacenter próprio.

Escalabilidade

Conforme visto na Aula 01, vamos agora fazer uso de uma das mais poderosas características da nuvem.

- Crie uma instância t2.micro com Ubuntu 18.04 LTS:
  - Renomeie a instância para *Webserver*
  - $ sudo apt update; sudo apt install nodejs build-essential -y
  - Edite um arquivo server.js:

```
#!/usr/bin/env nodejs
var http = require('http');
var os = require('os');
var crypto = require('crypto');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  var nonce = 1;
  var seed = Math.random();
  var h = crypto.createHash('sha256');

  h.update(new Buffer(nonce + " Hello World " + seed));
  while (h.digest("hex").substr(0,3)!='000') {
    h = crypto.createHash('sha256');
    nonce++;
    h.update(new Buffer(nonce + " Hello World " + seed));
  }

  res.end('{ "host": ' + os.hostname() + ', "nonce": ' + nonce + '}');
}).listen(8080, '');
console.log('Server running at http://localhost:8080/');
```

- Execute o serviço:
  - $ chmod +x ./server.js
  - $ ./server.js
- Libere a porta no *security group* correspondente. Teste o serviço no browser.
- Ajustando o final:
  - $ crontab -e
    @reboot /home/ubuntu/server.js
  - $ sudo reboot
- Teste novamente no browser.

1. O que faz o crontab?

Simulando carga no servidor

Uma das grande dúvidas de quem utiliza nuvem diz respeito a capacidade das instâncias utilizadas e se elas estão dimensiondas corretamente. Nessa etapa vamos estimar essa capacidade.

- Crie duas novas instâncias t2.micro com Ubuntu 18.04
- Renomeie como: Locust-master e Locust-slave
- Em ambas as máquinas:
  - $ sudo apt update; sudo apt install python3-pip -y
  - $ pip3 install locustio

  - crie um arquivo locustfile.py no home:

```
from locust import HttpLocust, TaskSet

def index(l):
    l.client.get("/")

class UserBehavior(TaskSet):
    tasks = {index: 1}

class WebsiteUser(HttpLocust):
    task_set = UserBehavior
    min_wait = 3000
    max_wait = 9000
```

- no master:
  - liberar no *security group* as portas 8080, 5557 e 5558
  - $ python3 -m locust.main –master –port=8080 –expect-slaves=1 –host=http://[IP Webserver]:8080
- No slave:
  - ajuste o número de conexões no Linux:
    $ sudo nano /etc/sysctl.conf
  - adicione a linha:
    fs.file-max = 5000000
  - aplique as alterações:
    $ sudo sysctl -p
  - acerte o crontab para rodar na inicialização:
    python3 -m locust.main –slave –master-host=[IP do master]
  - $ sudo reboot
- Realizando a simulação:
  - acesse o locust master via browser
  - aguarde o slave se conectar no master
  - inicie um teste 300/10 - 300 usuários e *hatch rate* de 10 novos usuários por segundo.
  - **Não caiu!** - Talvez haja algumas poucas falhas.
  - verifique que essa aplicação suporta cerca de 50 rps (requisições por segundo).
- Multiplicando os usuários:
  - crie uma imagem do locust-slave. Café.
  - crie mais 2 instâncias dessa AMI,
  - Execute um teste 600/20 - observe que há algumas falhas mas ainda em um percentual baixo.
  - Agora execute um teste 1000/50 - agora complicou.

1. O que é AMI?

Escalando Verticalmente

- pare a máquina com o servidor http.
- troque para t2.2xlarge.
- atualize o IP no locust master, reinicie os slaves e repita o teste 1000/50.

1. Quais as diferenças técnicas entre uma t2.micro e t2.2xlarge?

2. Por que a taxa de falhas ainda está alta?

Escalando Horizontalmente

- Crie uma imagem do servidor web. Café.
- Rebaixe a máquina para t2.micro de novo.
- Crie mais 2 instâncias a partir da AMI criada. Não esqueça de reutilizar o security group.
- Crie um *Application LoadBalancer* que mapeia instâncias:8080 para loadbalancer:80.
- Marcar todas as *Availability Zones*.
- Adicione as 3 instâncias no *Target Group*.
- Teste o acesso ao *LoadBalancer*:

  – Endereço DNS:

  – Verifique no navegador que há uma alternância do servidor.
  – Execute novamente o teste 1000/50.
  – Voilà!

1. Existe também limitações no *LoadBalancer* ou basta adicionar máquinas? Teste à vontade.

2. O que faz o *LoadBalancer*? Explique o algoritmo utilizado.

Agora vamos ativar a elasticidade da nuvem.

- Remova as instâncias do LoadBalancer. Destrue-as.
- Crie um *Autoscalling Group*:
  - Criar um *Launch Configuration* da mesma configuração em que foi criada a instância original. Reutilizar o *Security Group* com as devidas liberações.
  - Começar com 1 instância.
  - Selecionar todas as *subnets*.
  - Anexar o *Target Group* criado anteriormente.
  - Selecionar: *Use scaling policies to adjust the capacity of this group*. Usar *Request Count Per Target* em 3000 (50 rps x 60 segundos)
- Aguarde a primeira instância subir e teste o *LoadBalancer* de novo.

Execute o teste 1000/50 novamente. Em 3 minutos ele se ajustará sozinho à demanda. Encerre o teste e em 15 minutos irá desativar a capacidade ociosa automaticamente. Entregue o roteiro sem destruir nada.

1. Enfim, como funciona o *Autoscalling Group* da AWS?

2. Qual a diferença entre escalabilidade horizontal e escalabilidade vertical?

3. Qualquer serviço/aplicativo pode fazer uso da escalabilidade horizontal? Um banco de dados pode?

Questões Complementares

1. O que é um *VPS*? Qual a diferença entre uma instância e um *VPS*?

2. Defina *Platform as a Service* (PaaS) e *Software as a Service* (SaaS).

3. O modelo *LoadBalancer* possui um custo fixo elevado. Como você justificaria o uso e a configuração de um *LoadBalancer* para uma empresa?

Concluindo

1. Defina *Public Cloud*. Cite a principal vantagem e desvantagem.

2. Defina *Infrastructure as a Service* (Iaas).

3. Defina Escalabilidade.

Conclusão: Imagine como é o processo de alocação de uma instância. O que é realmente uma instância? Como você montaria um ambiente similar a AWS em um Datacenter próprio?