

Artigo

[Angelo Bruno Braga](#) · Mar. 7, 2022 7min de leitura

Executando o InterSystems IRIS em um modo FaaS utilizando o Kubeless

Function as a service (FaaS) é uma categoria de serviços de computação na nuvem que disponibiliza uma plataforma que permite que os clientes desenvolvam, executem e gerenciem as funcionalidades das aplicações sem que tenham a complexidade de construir e manter a infraestrutura tipicamente associada a se desenvolver e lançar um app. Construir uma aplicação seguindo este modelo é uma das formas de se alcançar uma arquitetura "serverless" e é tipicamente utilizada quando se constroem aplicações baseadas em microsserviços.

[Wikipedia](#)

FaaS é uma abordagem extremamente popular para se executar cargas de trabalho na nuvem, permitindo que os desenvolvedores mantenham o foco na escrita de códigos.

Este artigo irá mostrar-lhe como implantar métodos do InterSystems IRIS na forma FaaS.

Instale o Kubernetes

Primeiramente, instale o Kubernetes 1.16. Existem vários guias disponíveis então eu não os copiarei aqui. Para que vocês saibam, estou usando o [minicube](#). Com o minicube, para executar o kubernetes basta executar este comando:

```
minikube start --kubernetes-versionv1.16.1
```

Instale o kubeless

Em seguida vamos instalar o [kubeless](#). O kubeless é um framework serverless nativo do Kubernetes que permite que você implante pequenos trechos de código sem se preocupar com as questões relacionadas a infraestrutura. Ele aproveita os recursos do Kubernetes para fornecer dimensionamento automático, roteamento de API, monitoramento, solução de problemas e muito mais.

```
kubectl create ns kubeless
kubectl create -f https://github.com/kubeless/kubeless/releases/download/v1.0.8/kubeless-v1.0.8.yaml
kubectl get pods -n kubeless
```

A saída deve ser algo assim:

NAME	READY	STATUS	RESTARTS	AGE
kubeless-controller-manager-666ffb749-26vhh	3/3	Running	0	83s

Você também precisa instalar um cliente kubeless (na mesma instância onde estiver o kubectl). Você consegue ele [aqui](#). A instalação no Linux é tão simples quanto:

```
sudo install kubeless /usr/local/bin/kubeless
```

Testar o kubeless

Primeiramente vamos implantar uma função Python simples para verificar se o kubeless funciona.

Crie o test.py:

```
def hello(event, context):  
    return event['data']
```

Para ler mais sobre funções no kubeless, verifique [este documento](#), geralmente as funções aceitam dois argumentos - evento e contexto com estes dados:

```
event:  
  data:                                # Dados do evento  
  foo: "bar"                            #  
Os dados são analisados como JSON quando solicitados  
  event-id: "2ebb072eb24264f55b3fff"    # ID do evento  
  event-type: "application/json"        # Tipo de conteúdo do evento  
  event-time: "2009-11-10 23:00:00 +0000 UTC" # Timestamp da origem do evento  
  event-namespace: "kafkatriggers.kubeless.io" # Emissor do evento  
  extensions:                            # Parâmetros opcionais  
    request: ...                          # Referência à requisição recebida  
    response: ...                          #  
Referência à resposta que deverá ser enviada  
  
    # (propriedades específicas dependerão da linguagem da função)  
context:  
  function-name: "pubsub-nodejs"  
  timeout: "180"  
  runtime: "nodejs6"  
  memory-limit: "128M"
```

Agora podemos implantar nossa função "hello" especificando nosso arquivo com a função e um ambiente de tempo de execução:

```
kubeless function deploy hello --runtime python3.7 --from-  
file test.py --handler test.hello  
kubeless function ls hello
```

E vamos testá-lo:

```
kubeless function call hello --data 'Hello world!'
```

Você deverá receber Hello World! como resposta.

Adicione a configuração IRIS

A seguir precisamos adicionar um manipulador de funções InterSystems IRIS, para fazê-lo, abra a configuração do kubeless para editá-la:

```
kubeless get-server-config  
kubectl get -n kubeless configmaps -o yaml > configmaps.yaml  
kubectl edit -n kubeless configmaps
```

Adicione esta entrada no array de imagens de runtime e grave:

```
{"ID": "iris", "depName": "", "fileNameSuffix": ".cls", "versions": [{"images": [{"image": "eduard93/kubeless-iris-runtime:latest", "phase": "runtime"}], "name": "iris2022.1", "version": "2022.1"}]}
```

Reinicie o controller do kubeless para que as alterações passem a funcionar.

```
kubectl delete pod -n kubeless -l kubeless=controller
```

Construa a função CRD IRIS e a publique

Agora vamos escrever nossa primeira função no InterSystems IRIS:

```
Class User.Test {  
  
ClassMethod hi(event, context) As %Status  
{  
    if $isObject(event) {  
        write event.Text + event.Text  
    } else {  
        write "HELLO FROM IRIS"  
    }  
    quit $$$OK  
}  
}
```

Depois precisamos criar uma função CRD:

Aqui está nosso modelo:

```
function.yaml
```

E precisamos preencher:

- name: nome da função (para o kubeless)
- handler: classe.nome do método (para o InterSystems IRIS)

- function : adicione no final o corpo da função (não esqueça dos tabs!)

Desta forma nossa CRD ficará assim:

```
functiondemo.yaml
```

Isso pode ser facilmente automatizado. No Linux execute:

```
sed 's/!name!/iris-  
demo/; s/!handler!/User_Test.hi/' function.yaml > function_demo.yaml  
sed 's/^/      /' User.Test.cls >> function_demo.yaml
```

E no Windows (PowerShell):

```
Get-Content function.yaml | ForEach-Object { $_ -replace "!handler!", "User_Test.hi"  
-replace "!name!", "iris-demo" } | Set-Content function_demo.yaml  
"      " + [string]((Get-Content User.Test.cls) -join "`r`n      ") | Add-  
Content function_demo.yaml
```

Agora precisamos publicar nossa CRD no kubeless:

```
kubectl apply -f function_demo.yaml
```

Testar a função IRIS

Primeiramente, vamos ver se a função está implantada e pronta (pode levar alguns minutos na primeira vez):

```
kubeless function ls
```

E agora chame ela:

```
kubeless function call iris-demo --data '{"Text":123}'
```

Se você estiver usando o Windows, chame a função assim (será o mesmo para todas as outras chamadas, com aspas duplas escapadas):

```
kubeless function call iris-demo --data '{"Text\":123}'
```

De qualquer forma, a resposta deverá ser 456 pois 123 é um número.

Acesso HTTP

O kubeless também disponibiliza acesso HTTP. Para testar, use o comando kubectl proxy:

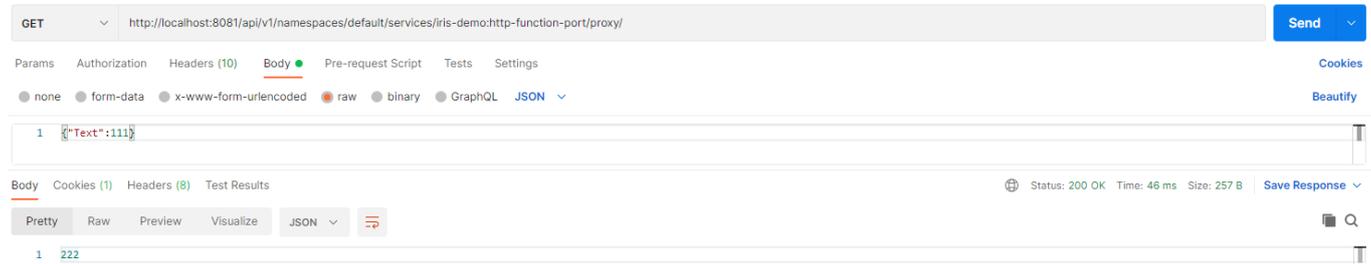
```
kubectl proxy -p 8081
```

Em seguida, envie esta requisição utilizando seu cliente preferido de APIs REST:

```
GET http://localhost:8081/api/v1/namespaces/default/services/iris-demo:http-function-port/proxy/
```

```
{"Text":111}
```

Aqui está como fica utilizando o Postman:



Vamos agora publicar na internet.

Existem duas abordagens. Preferencialmente configure o ingress conforme descrito [aqui](#).

Adicionalmente você pode corrigir o serviço da função:

```
kubectl get svc
kubectl patch svc iris-demo -p '{"spec": {"type": "LoadBalancer"}}'
kubectl get svc
```

Limpendo

Para remover uma chamada da função implantada:

```
kubectl delete -f function_demo.yaml
```

Conclusão

Enquanto esta é sem dúvida uma prova de conceito e não uma solução a nível de produção, esta abordagem demonstra que é possível executar cargas de trabalho do InterSystems IRIS utilizando a abordagem serverless FaaS.

Links

- [Minicube](#)
- [Kubeless](#)
- [InterSystems IRIS runtime](#)

[#Docker](#) [#Nuvem](#) [#InterSystems IRIS](#)

URL de origem: <https://pt.community.intersystems.com/post/executando-o-intersystems-iris-em-um-modo-faas-utilizando-o-kubeless>
