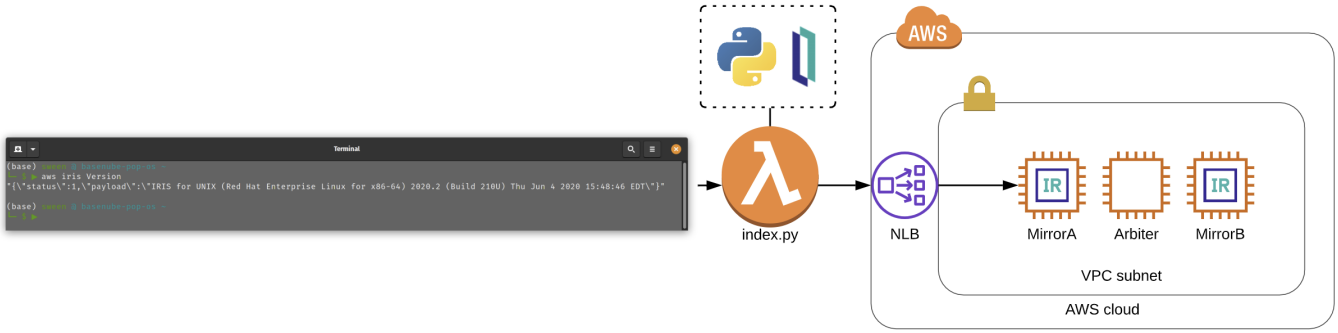
Artigo
Ron Sweeney · Fev. 10, 2021 7min de leitura

API Nativa IRIS para Python na AWS Lambda

Se você está procurando uma maneira inteligente de integrar sua solução IRIS no ecossistema Amazon Web Services, aplicativos sem servidor ou script python baseado em boto3, usar a API Nativa IRIS para Python pode ser o caminho a seguir. Até que você precise obter ou definir algo no IRIS, você não tem que ir muito longe com uma implementação em produção para fazer sua aplicação funcionar de um maneira incrível, então, espero que você encontre valor neste artigo e construa algo que seja importante para outros ou somente para você, pois ambos são igualmente válidos.



Se você está procurando algumas justificativas para implementar isso:

- Você precisa acertar a trigger de geração de pré-token no Cognito para pesquisar e inserir o contexto de ID do paciente no token para uma solução baseada em SMART on FHIR(R) implementando um fluxo de trabalho com OAUTH2.
- Você deseja publicar as configurações de ajuste de provisionamento do IRIS com base no tipo de instância, grupo de nós, toaster ou cluster ECS que você disparou para executar o IRIS no anel zero.
- Você quer impressionar a família e os amigos no Zoom com suas habilidades de gerenciamento do IRIS sem que o seu shell saia da AWS cli.

O ponto

Aqui, vamos provisionar uma função lambda da AWS que se comunica com o IRIS e fornecer alguns exemplos sobre como provisioná-la e como interagir com ela em várias capacidades, na esperança de que possamos discutir sobre isso e **publicá-la no pip para tornar as coisas mais fáceis.**

Com pressa????

Confira o streaming

https://www.youtube.com/embed/mA0VzKOYhBk
*[Isso é um link incorporado, mas você não pode ver conteúdo incorporado diretamente no site, porque recusou os cookies necessários para acessá-lo. Para ver o conteúdo incorporado, você precisa aceitar todos os cookies nas suas **Definições de cookies**]*

Em todo o caso...

Para participar da diversão, você precisará definir algumas coisas em seu plano de voo.

Rede

Você tem o IRIS em execução da maneira que deseja, rodando em uma AWS VPC com exatamente duas sub-redes e um grupo de segurança que permite acesso ao super servidor rodando IRIS... usamos 1972 por motivos nostálgicos e pelo simples fato de que a InterSystems se deu ao trabalho de registrar essa porta no IANA, e se você adicionar a nova porta em /etc/services sem fazer isso, não é a melhor prática, mas tudo bem. Em nosso caso, é um conjunto de instâncias EC2 espelhadas com verificações de integridade adequadas em torno de um balanceador de carga de rede AWS v2.

Classe de exemplo importada

De alguma forma, você conseguiu criar e importar a classe na raiz deste repositório para o namespace %SYS em sua instância IRIS. A seguir está um exemplo da classe que gera a saída acima. Se você está se perguntando por que precisamos importar uma classe aqui, consulte a nota abaixo, onde a abordagem recomendada é provisionar algumas classes de wrapper para uso por meio do python.

> Nota do Docs: Embora esses métodos também possam ser usados com as classes InterSystems definidas na Biblioteca de Classes, a melhor prática é chamá-las indiretamente, de dentro de uma classe ou rotina definida pelo usuário. Muitos métodos de classe retornam apenas um código de status, passando os resultados reais de volta em um argumento (que não pode ser acessado pela API nativa). As funções definidas pelo sistema (listadas em Funções ObjectScript na Referência ObjectScript) não podem ser chamadas diretamente.

Classe de Exemplo:

```
Class ZDEMO.IRIS.Lambda.Operations Extends %Persistent
{
  ClassMethod Version() As %String
  {
   Set tSC = 0

   Set tVersion = $ZV
   if ( tVersion '="" ) { set tSC = $$$OK }

   Set jsonret = {}
   Set jsonret.status = tSC
   Set jsonret.payload = tVersion

   Quit jsonret.%ToJSON()
  }
}
```

Observe acima que decidi trabalhar com base na convenção de sempre retornar um objeto JSON como uma resposta, que também me permite retornar o status e possivelmente preencher a lacuna ao retornar algumas coisas por referência.

Acesso AWS

Obtenha algumas chaves de acesso IAM que permitirão que você provisione e invoque a Função Lambda com a qual iremos mudar o mundo.

Verificação antes do voo:

```
IRIS            [ $$$OK ]
VPC             [ $$$OK ]
Subnets         [ $$$OK ]
Security Group  [ $$$OK ]
IAM Access      [ $$$OK ]
Imported Class  [ $$$OK ]
```

$$$OK, **Vamos lá**.

Empacotando a API Nativa IRIS para Python para uso na Função Lambda

Esta é a parte que seria fantástica se fosse um pacote pip, especialmente se fosse apenas para Linux, já que as funções da AWS Lambda são executadas no BoXenLinux. No momento deste commit, a API não está disponível via pip, mas somos engenhosos e podemos lançar a nossa própria.

```
mkdir iris_native_lambda
cd iris_native_lambda
wget https://github.com/intersystems/quickstarts-python/raw/master/Solutions/nativeAPI_wheel/irisnative-1.0.0-cp34-abi3-linux_x86_64.whl
unzip nativeAPI_wheel/irisnative-1.0.0-cp34-abi3-linux_x86_64.whl
```

Crie connection.config

Exemplo: connection.config

Crie seu manipulador, index.py ou use aquele na pasta de exemplos, no repositório de demonstração no GitHub. Observe que a versão de demonstração usa variáveis de ambiente e um arquivo externo para as informações de conectividade IRIS.

Exemplo:
index.py

Agora compacte-o para uso:

```
zip -r9 ../iris_native_lambda.zip *
```

Crie um bucket S3 e carregue o zip de função nele.

```
cd ..
aws s3 mb s3://iris-native-bucket
s3 sync iris_native_lambda.zip s3://iris-native-bucket
```

Isso conclui o empacotamento da API e do manipulador para uso como uma função AWS Lambda.

Agora, clique no console para criar a função ou use algo como o Cloudformation para executar o trabalho:

```
  IRISAPIFunction:
    Type: "AWS::Lambda::Function"
    DependsOn:
      - IRISSG
      - VPC
    Properties:
      Environment:
        Variables:
          IRISHOST: "172.31.0.10"
          IRISPORT: "1972"
          NAMESPACE: "%SYS"
          USERNAME: "intersystems"
          PASSWORD: "lovetheyneighbor"
      Code:
        S3Bucket: iris-native-bucket
        S3Key: iris_native_lambda.zip
      Description: "Função API Nativa IRIS para Python"
      FunctionName: iris-native-lambda
      Handler: "index.lambda_handler"
      MemorySize: 128
      Role: "arn:aws:iam::8675309:role/BeKindtoOneAnother"
      Runtime: "python3.7"
      Timeout: 30
      VpcConfig:
        SubnetIds:
          - !GetAtt
            - SubnetPrivate1
            - Outputs.SubnetId
          - !GetAtt
            - SubnetPrivate2
            - Outputs.SubnetId
        SecurityGroupIds:
          - !Ref IRISSG
```

Isso foi BASTANTE, mas agora você pode enlouquecer e chamar o IRIS através da função lambda com Python e mudar o mundo.

Execução!

Da forma como o descrito acima é implementado, espera-se que a função seja passada em um objeto de evento que é um tanto estruturado para reutilização, você pode ver a ideia no exemplo de objeto de evento abaixo:

```
{
  "method": "Version",
  # importante, se o método não requer argumentos, aplique "none"
  "args": "none"
  # exemplo de método com argumentos, separados por vírgulas
  # "args": "thing1, thing2"
}
```

agora você pode, se tiver tolerância para exemplos em linha de comando, dar uma olhada na execução abaixo usando a AWS CLI:

```
(base) sween @ basenube-pop-os ~/Desktop/BASENUBE
?? $ &#x25b6; aws lambda invoke --function-name iris-native-lambda --payload '{"metho
d":"Version","args":"none"}' --invocation-type RequestResponse --cli-binary-
format raw-in-base64-out --region us-east-2 --profile default /dev/stdout
{{\"status\":1,\"payload\":\"IRIS for UNIX (Red Hat Enterprise Linux for x86-64) 2020
.2 (Build 210U) Thu Jun 4 2020 15:48:46 EDT\"}"
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

Agora, se dermos um passo adiante, a AWS CLI oferece suporte a aliases, então crie um para você mesmo e você pode brincar com total integração com o seu comando aws legal. Aqui está um exemplo de alias cli:

```
?? $ &#x25b6; cat ~/.aws/cli/alias
[toplevel]
whoami = sts get-caller-identity

iris =
  !f() {
   aws lambda invoke \
     --function-name iris-native-lambda \
     --payload \
     "{\"method\":\""${1}"\",\"args\":\"none\"}" \
     --invocation-type RequestResponse \
     --log-type None \
     --cli-binary-format raw-in-base64-out \
     gar.json > /dev/null
     cat gar.json
     echo
     echo
  }; f
```

...e agora, você pode apenas fazer...

Se cuide! – Argumentos técnicos são bem-vindos!

#InterSystems IRIS #InterSystems IRIS for Health

URL de origem:https://pt.community.intersystems.com/post/api-nativa-iris-para-python-na-aws-lambda