

Manual utilização Cloud Foundry (MindSphere)

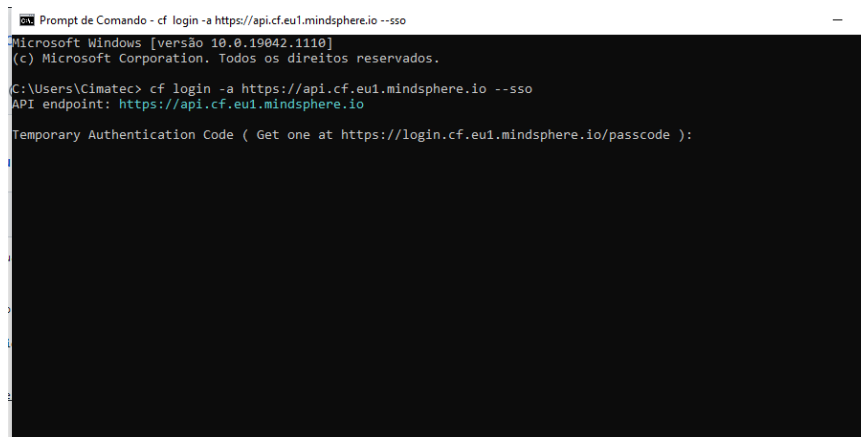
Sumário

1. Primeiros passos
 - Conectando-se ao cloud foundry
2. Configurando conta e/ou ambiente
3. Deploy de aplicação
 - Configuração manifest.yml
 - Publicando no Cloud Foundry
4. Serviços
 - Criando um serviço
 - Criando as chaves do serviço
 - Acessando serviços utilizando SSH
5. Referências

Primeiros passos

Conectando com o cloud foundry

No terminal utilize o comando `cf login -a https://api.cf.eu1.mindsphere.io --sso`



```
Prompt de Comando - cf login -a https://api.cf.eu1.mindsphere.io --sso
Microsoft Windows [versão 10.0.19042.1110]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Cimatec> cf login -a https://api.cf.eu1.mindsphere.io --sso
API endpoint: https://api.cf.eu1.mindsphere.io

Temporary Authentication Code ( Get one at https://login.cf.eu1.mindsphere.io/passcode ):
```

Fig.1 - Terminal do cf.

Se estiver tudo certo vai aparecer um link para geração de token de autenticação. Esse link deve ser colado no navegador.

The image shows the Siemens login interface. At the top is the 'SIEMENS' logo in teal. Below it is the heading 'Verificar sua identidade' (Verify your identity). A subtext says 'Enviamos um e-mail com o código para' (We send an email with the code to). There is a grey rectangular box for email input. Below that is a blue-bordered box labeled 'Inserir o código' (Enter the code) with a cursor. An orange button labeled 'Continuar' (Continue) is below the code box. At the bottom, there is a link 'Não recebeu um e-mail? Reenviar' (Didn't receive an email? Resend) and a link 'Você não? Sair' (You don't? Logout).

Fig.2 - Tela de verificação de identidade.

Após inserir o código é gerado um token que deve ser adicionado a requisição no terminal.

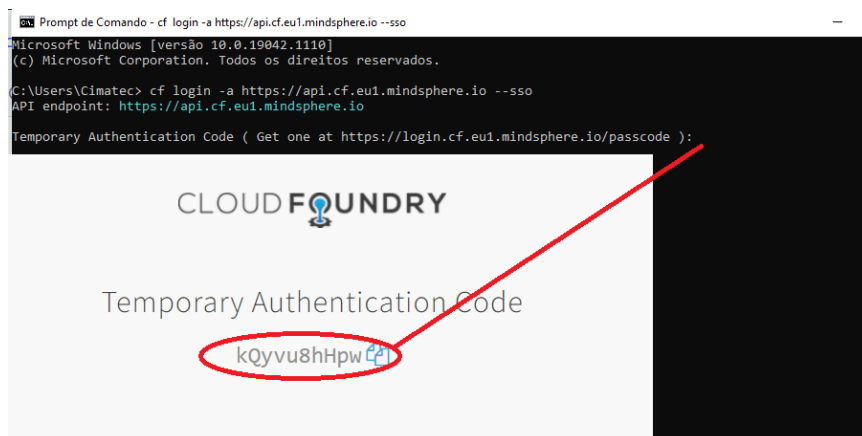


Fig.3 - Token de autenticação.

Tudo certo! Se o código for validado com sucesso você já está autorizado para acessar a plataforma.

Se tudo der certo o retorno será equivalente:

```
API endpoint:  https://api.cf.eu1.mindsphere.io
API version:   3.107.0
user:         SEU USUÁRIO
org:          ORG DA SUA EMPRESA
space:        SPACE DA SUA EMPRESA
```

Configurando conta e/ou ambiente

Caso sua org e/ou space não estejam disponíveis ou não estiver autorizado para fazer o push de uma aplicação

Para dar o deploy e gerenciar suas aplicações é preciso que a conta esteja atrelada a um org e space (para mais informações acesse a sessão de referências). Para criar uma Org e Space use os seguintes comandos (Para executar esse comando é preciso ter perfil administrador).

```
cf create-org ORG
```

```
cf create-space SPACE [-o ORG] [-q SPACE_QUOTA]
```

Onde "ORG" e "SPACE" é o nome que deseja dar a Org e o Space respectivamente.

Para dar o push de uma aplicação é preciso ter a autorização SpaceDeveloper. Para isso use os seguintes comandos:

```
cf set-org-role USERNAME ORG OrgManager
cf set-space-role USERNAME ORG SPACE SpaceDeveloper
```

Onde "USERNAME" é o usuário que deseja atribuir a autorização.

Pronto! Agora o usuário está apto para dar o deploy de uma aplicação.

Deploy de aplicação

Antes de continuar assegure-se de ter uma conta com perfil SpaceDeveloper. Utilize o comando `cf space-users ORG SPACE`.

A aplicação utilizada para deploy foi desenvolvida em Spring Boot.

Gerando arquivo .war

Para gerar o arquivo .war, usando o STS, acessar "File->Export-> Selecionar Web -> Clicar em WAR File"

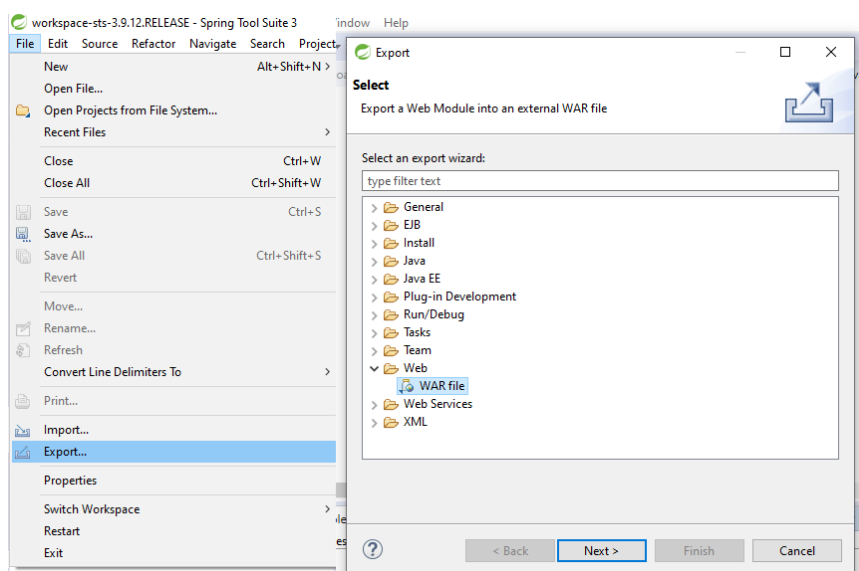


Fig.4 - Geração de arquivo .War no STS.

Configuração manifest.ymlm

Essa aplicação utiliza o serviço do Postgresql para mais informações visite a sessão [Serviços](#)

A configuração básica do manifest.ymlm para o projeto é:

```

applications:
- name: NOMEAPLICAÇÃO
  instances: 1
  buildpacks:
  - java_buildpack
  path: ./APP.war #Caminho da aplicação
  random-route: true
  memory: 1024MB #Quantidade de memória ram
  disk_quota: 500MB #Quantidade de espaço disponível para a aplicação
  services:
  - NOMESERVIÇO #Nome do serviço criado

```

Publicando no Cloud Foundry

Finalizando as configurações basta apenas utilizar o comando `cf push`

Serviços

O serviço utilizado será o do PostgreSQL

Criando um serviço

Para criar um serviço primeiramente é preciso saber os serviços disponíveis para uso, utilize o seguinte comando:

```
cf marketplace
```

offering	plans	description
postgresql94	postgresql-m, postgresql-xs, postgresql-m50, postgresql-m150_ha, postgresql-s10, postgresql-s10_ha	broker
		Dedicated PostgreSQL service instances and clusters powered by the anynines Service Framework
rabbitmq36	rabbitmq-m, rabbitmq-xs, rabbitmq-m50, rabbitmq-m150_ha, rabbitmq-s10, rabbitmq-s10_ha	a9s-postgresql-broker
		Dedicated RabbitMQ service instances powered by the anynines Service Framework
logme	logme-m, logme-xs, logme-m150_ha, logme-m50, logme-s10, logme-s10_ha	a9s-rabbitmq-broker
		Dedicated ELK stacks to monitor applications and service instances powered by the anynines Service Framework
elasticsearch6	elasticsearch-m, elasticsearch-xs, elasticsearch-m50, elasticsearch-m150_ha, elasticsearch-s10, elasticsearch-s10_ha	a9s-logme-broker
		Dedicated Elasticsearch 6.x service instances and clusters powered by the anynines Service Framework
redis32	redis-m, redis-xs, redis-m50, redis-m150_ha, redis-s10, redis-s10_ha	a9s-elasticsearch-broker
		Dedicated Redis service instances and clusters powered by the anynines Service Framework
redis40	redis-m, redis-xs, redis-m150_ha, redis-m50, redis-s10, redis-s10_ha	a9s-redis-broker
		Dedicated Redis service instances and clusters powered by the anynines Service Framework
rabbitmq37	rabbitmq-m, rabbitmq-xs, rabbitmq-m50, rabbitmq-m150_ha, rabbitmq-s10, rabbitmq-s10_ha	a9s-redis-broker
		This is a service creating and managing dedicated RabbitMQ service instances, powered by the anynines Service Framework
postgresql10	postgresql-m, postgresql-xs, postgresql-m50, postgresql-m150_ha, postgresql-s10, postgresql-s10_ha	a9s-rabbitmq-broker
		This is a service creating and managing dedicated PostgreSQL service instances and clusters, powered by the anynines Service Framework
elasticsearch6	elasticsearch-m, elasticsearch-xs, elasticsearch-m50, elasticsearch-m150_ha, elasticsearch-s10, elasticsearch-s10_ha	a9s-postgresql-broker
		This is the anynines Elasticsearch 6.x service
postgresql11	postgresql-m, postgresql-xs, postgresql-m50, postgresql-m150_ha, postgresql-s10, postgresql-s10_ha	a9s-elasticsearch-broker
		This is a service creating and managing dedicated PostgreSQL service instances and clusters, powered by the anynines Service Framework
redis50	redis-m, redis-xs, redis-m50, redis-m150_ha, redis-s10, redis-s10_ha	a9s-postgresql-broker
		This is a service creating and managing dedicated Redis service instances, powered by the anynines Service Framework
rabbitmq38	rabbitmq-m, rabbitmq-xs, rabbitmq-m50, rabbitmq-m150_ha, rabbitmq-s10, rabbitmq-s10_ha	a9s-redis-broker
		This is a service creating and managing dedicated Messaging service instances, powered by the anynines Service Framework
		a9s-rabbitmq-broker

Fig.5 - Serviços disponíveis no marketplace do cf.

Para o projeto será utilizado o serviço postgresql94 com o plano postgresql-m. Para criar um serviço no seu Space utilize:

```
cf create-service SERVIÇO PLANO NOMESERVIÇO
```

Para o projeto ficaria o seguinte: `cf create-service postgresql94 postgresql-m postgresql-carsharing`.

Antes de utilizar o serviço aguarde alguns minutos até que o status seja atualizado para "create succeeded" para verificar utilize o comando `cf services` e localize o serviço criado.

name	offering	plan	bound apps	last operation	broker	upgrade available
postgresql-carsharing	postgresql94	postgresql-m	carsharing	create succeeded	a9s-postgresql-broker	no
postgresql-eletrposto	postgresql94	postgresql-m	eletrposto	update succeeded	a9s-postgresql-broker	no

Fig.6 - Criação de serviço.

Criando as chaves do serviço

Para o caso do projeto é preciso criar chaves para acesso ao banco do Postgres:

```
cf create-service-key NOME DOSERVIÇO CRIADO NOME DAKEY
```

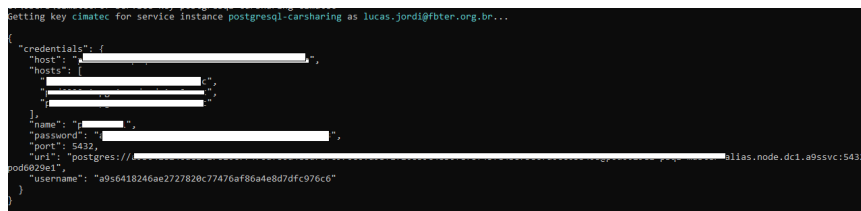
Exemplo :

```
cf create-service-key postgresql-carsharing cimaterc
```

Se a key foi criada com sucesso basta executar o comando:

```
cf service-key NOME DOSERVIÇO NOME DAKEY
```

A resposta deve ser algo como:



```
Setting key cimaterc for service instance postgresql-carsharing as lucas.jordi@fbter.org.br...
{
  "credentials": {
    "host": "ueueue-psql-master-alias.node.dcl.sjusu",
    "hosts": [
      "ueueue-psql-master-alias.node.dcl.djndh",
      "ueueue-psql-master-alias.node.dcl.sjiidiusu",
      "ueueue-psql-master-alias.node.dcl.ddudu"
    ],
    "name": "ususuuusu",
    "password": "viidjffjidfijfdjifd",
    "port": 5432,
    "uri": "postgres://fijsgdsfiogdiogjdoigd:jidsfojofdjisdojifdjoidf@dfiudi-psql-master-alias.node.dcl.a9ssvc:5432/ususuuusu",
    "username": "fdsuufidudfjufsdsdufi"
  }
}
```

Fig.7 - Chaves de acesso serviço do postgresql.

Agora é só adicionar o serviço no manifest.yml e as keys na sua aplicação para poder utilizar o serviço.

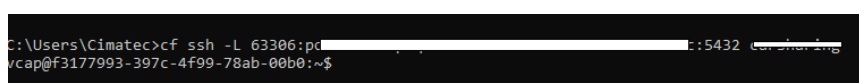
Acessando serviços utilizando SSH

```
cf ssh -L PORTA:HOSTNAME:PORTA DOSERVIÇO NOME DO APP QUE UTILIZA O SERVIÇO
```

Digamos que tenha sido criado um serviço chamado serviço1 que é utilizado pela aplicação 1 cadastrada no sistema como app1 cujas keys são:

```
{
  "credentials": {
    "host": "ueueue-psql-master-alias.node.dcl.sjusu",
    "hosts": [
      "ueueue-psql-master-alias.node.dcl.djndh",
      "ueueue-psql-master-alias.node.dcl.sjiidiusu",
      "ueueue-psql-master-alias.node.dcl.ddudu"
    ],
    "name": "ususuuusu",
    "password": "viidjffjidfijfdjifd",
    "port": 5432,
    "uri": "postgres://fijsgdsfiogdiogjdoigd:jidsfojofdjisdojifdjoidf@dfiudi-psql-master-alias.node.dcl.a9ssvc:5432/ususuuusu",
    "username": "fdsuufidudfjufsdsdufi"
  }
}
```

Então o comando seria `cf ssh -L 63306:ueueue-psql-master-alias.node.dcl.sjusu:5432 app1`



```
C:\Users\Cimatec>cf ssh -L 63306:ueueue-psql-master-alias.node.dcl.sjusu:5432 app1
vcap@f3177993-397c-4f99-78ab-00b0:~$
```

Fig.8 - Resultado de conexão com o container usando SSH.

Imagem mostra o resultado da conexão estabelecida.

Agora se quisermos acessar o banco utilizamos o serviço do Postgresql instalado na máquina. Para esse caso foi utilizado o PhpAdmin.

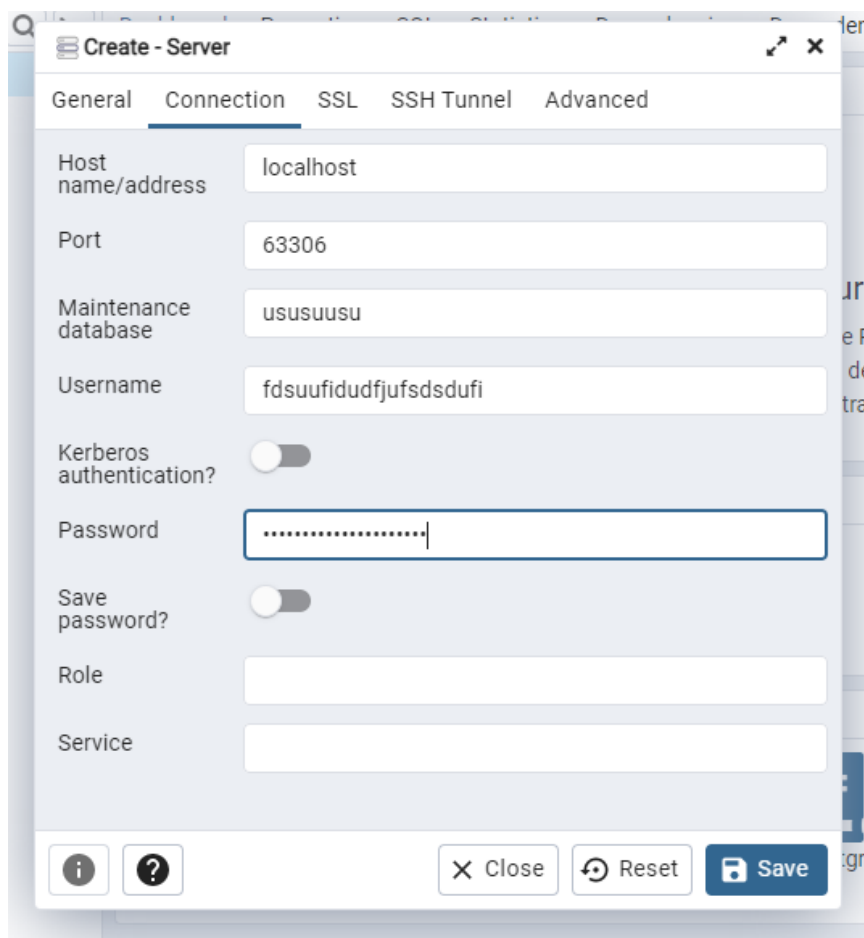


Fig.9 - Conexão do banco usando o PhpMyAdmin.

Lembrar de colocar a porta escolhida no SSH e não a que está na key. Nesse caso é a 63306.

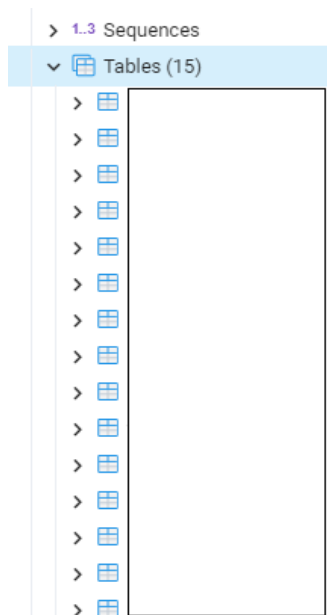


Fig.10 - Tabelas do banco cadastrado no cf.

É possível ver que as tabelas batem com as do projeto.

Referências

[Serviços e Keys](#)

[Conectando serviços com SSH](#)

[Perfis de autorização](#)

[Deploy Spring Apps](#)

[Publicando apps](#)