Deploy para CrewAI

Asimov Academy

ASIMOV

Conteúdo

01. Por que o Deploy é Importante? 3
Visão Geral . . . 3
Próximos passos . . . 3

02. Criando um projeto pronto para deploy 4
Acessando o CrewAI Enterprise . . . 4
Configuração do Ambiente e Criação do Projeto . . . 4
Preparação do Ambiente Virtual . . . 4
Criação do Projeto com o `crewai create crew` . . . 5
Execução do Projeto . . . 5
Considerações Finais . . . 6

03. Adicionando meu projeto ao GitHub 7
Preparando o Ambiente no VS Code . . . 7
Inicializando o Repositório Git . . . 7
Configurando o Arquivo .gitignore . . . 7
Importância do .gitignore . . . 7
Fazendo o Primeiro Commit . . . 8
Sincronizando o Repositório Local com o GitHub . . . 8
Adicionando o Remote no VS Code . . . 8
Realizando o Push para o GitHub . . . 9
Conclusão e Próximos Passos . . . 9

04. Fazendo deploy de uma crew 10
Acessando o CrewAi Enterprise . . . 10
Configurando Variáveis de Ambiente . . . 10
Realizando o Deploy . . . 11
Considerações Finais e Próximos Passos . . . 11

05. Testando a API com Postman 13
Postman . . . 13
Preparando a Requisição . . . 13
Teste Inicial da API . . . 13
Iniciando o Processo com Kickoff . . . 14
Exemplo de Corpo da Requisição . . . 14
Monitorando a Execução . . . 14
Conclusão . . . 14

06. Testando a API com Python **16**
Preparando o ambiente 16
Consultando os inputs 17
Iniciando uma execução (kickoff) 17
Acompanhando o status da execução 17
Conclusão 18

01. Por que o Deploy é Importante?

De que adianta desenvolver um sistema multi-agentes incrível se ninguém consegue acessá-lo? O deploy é o passo que transforma sua aplicação em algo acessível: seja via web, APIs ou aplicativos, permitindo que qualquer pessoa, em qualquer lugar do mundo, interaja com sua solução.

Visão Geral

Neste breve curso de Deploy para CrewAI, abordaremos o caminho mais simples e intuitivo para colocar sua Crew em produção:

1. **Criar um projeto pronto para deploy**: estrutura de arquivos e configuração inicial.
2. **Adicionar o projeto ao GitHub**: versionamento e integração contínua.
3. **Conectar o GitHub ao CrewAI Enterprise**: automação do processo de build e deploy.
4. **Executar o deploy**: publicando a API criada, testando via Postman e integrando em código Python.

Ao final, você terá um projeto completo — da criação à implantação — e entenderá em profundidade como gerar uma API a partir das suas Crews.

Próximos passos

No próximo capítulo vamos detalhar como configurar o `main.py` para inicializar sua Crew AI, definindo endpoints e preparando a aplicação para receber requisições.

02. Criando um projeto pronto para deploy

Nesta aula, iremos aprender como criar um projeto com o CrewAI e prepará-lo para o deploy. O deploy será realizado utilizando o **CrewAI Enterprise**, uma solução criada pela comunidade dos desenvolvedores do CrewAI para a realização de deploys específicos para aplicativos construídos com CrewAI.
Atenção: A plataforma não suporta deploy para outras tecnologias, como Streamlit ou Dash.

Acessando o CrewAI Enterprise

1. **Login e Integração com o GitHub:**
    - Acesse a interface pelo endereço: www.crewai.com/enterprise
    - Realize o login na plataforma.
    - Conecte sua conta do GitHub, pois o CrewAI Enterprise necessita de acesso aos seus repositórios.
    - **Observação:** Se o projeto não estiver hospedado no GitHub, o deploy não será possível.
2. **Fluxo de Deploy Utilizando o GitHub:**
    - Durante o curso, veremos não apenas a criação do projeto mas também o processo de publicação no GitHub.
    - Em seguida, aprenderemos a criar uma API a partir desse projeto utilizando o CrewAI Enterprise.

Configuração do Ambiente e Criação do Projeto

Preparação do Ambiente Virtual

Antes de criar o projeto, é importante garantir que o ambiente virtual está configurado corretamente, conforme os requisitos de versão do Python:

- **Versões Suportadas:** O CrewAI funciona melhor com Python nas versões 3.10 até 3.12.
- **Criação e Ativação do Ambiente Virtual:**
- **Linux:**

```
source ./venv/bin/activate
```

- Windows (PowerShell):

```
.\venv\Scripts\Activate.ps1
```

Criação do Projeto com o `crewai create crew`

1. **Inicializando o Projeto:**
   Com o ambiente virtual ativo e o CrewAI Tools instalado, utilize o comando de criação de projeto. Por exemplo, para um projeto de deploy, execute:

```
crewai create "Crew for Deploy"
```

- Esse comando gera a estrutura básica do projeto, contendo arquivos como **pyProject**, **README.md** e o arquivo **.env** para configuração de variáveis.

2. **Configuração da Chave de API:**
   - Abra o arquivo **.env** e insira sua chave de API da OpenAI.
   - Certifique-se de que a chave está corretamente atribuída para que o projeto funcione em conjunto com os serviços da OpenAI.

3. **Observação sobre Modificações no Projeto:**
   - Diferentemente de outros experimentos com o CrewAI, não serão necessárias alterações manuais nos arquivos como o **main.py** ou a adição de componentes como o **AgentOps**.
   - Utilize o projeto exatamente como criado pelo comando `crewai create`, pois ele já está preparado para o deploy.

Execução do Projeto

Após a criação do projeto e configuração do ambiente, execute-o para verificar se tudo está funcionando corretamente:

1. **Navegue até o Diretório do Projeto:**

```
cd "Crew for Deploy"
```

2. **Executando o Projeto:**

Utilize o comando específico do CrewAI para rodar o projeto:

```
crewai run
```

- Durante essa execução, o sistema cria automaticamente um novo ambiente virtual (se necessário) e configura as dependências do projeto.
  - Um arquivo de lock, por exemplo, `uv.lock`, é gerado. Esse arquivo é essencial para que o servidor de deploy saiba quais versões das bibliotecas devem ser instaladas, garantindo que o ambiente replicado seja idêntico ao utilizado durante o desenvolvimento.

3. **Validação:**
   Se o projeto executa sem erros após o comando `crewai run`, significa que a estrutura está correta e pronta para o deploy.

Considerações Finais

- **Setup do Projeto:**
  O setup realizado nesta aula assegura que seu projeto esteja devidamente configurado e preparado para as etapas seguintes:
  - Publicação no GitHub.
  - Realização do deploy via CrewAI Enterprise.
- **Próximos Passos:**
  A seguir, veremos como adicionar o projeto criado ao GitHub. Em seguida, abordaremos o processo completo para realizar o deploy da aplicação.

03. Adicionando meu projeto ao GitHub

Nesta aula, vamos aprender como adicionar o projeto (já criado e preparado para deploy) ao GitHub. Essa etapa é fundamental porque o **CrewAI Enterprise** utilizará o repositório hospedado no GitHub para acessar o seu código e realizar o deploy. Embora a abordagem mostrada utilize o Visual Studio Code (VS Code), você pode adotar outras ferramentas como Git Bash, GitHub Desktop ou a linha de comando, conforme sua preferência.

Preparando o Ambiente no VS Code

- **Contexto:**
  Durante a criação do projeto com o comando de criação do CrewAI, foi gerado um diretório (neste exemplo, chamado "`Crew for Deploy`") que contém tudo que é necessário para o deploy.
- **Ação:**
  Abra a pasta do projeto diretamente no VS Code.
  Dica: Garanta que você esteja trabalhando **somente** com os arquivos que estão dentro desta pasta, descartando elementos que estejam fora dela.

Inicializando o Repositório Git

- No VS Code, acesse a área de **Controle de Código Fonte** e clique em **Initialize Repository** para transformar sua pasta em um repositório Git local.
- Após a inicialização, verifique se os arquivos do projeto aparecem na lista de mudanças pendentes (staged ou unstaged).

Configurando o Arquivo .gitignore

Importância do .gitignore

O arquivo **.gitignore** é essencial para evitar que arquivos ou pastas indesejadas sejam enviados para o repositório remoto. No seu caso:

- **Ambiente Virtual:**
  Evite subir a pasta do ambiente virtual (por exemplo, `.venv` ou `venv`). Lembre-se de que o sistema de deploy criará um novo ambiente virtual com base no arquivo de dependências (como o `uv.lock`).

- **Variáveis de Ambiente:**
  O arquivo **.env** contém segredos, como a chave da API da OpenAI. Não faça o commit deste arquivo para o repositório público. ### Exemplificação

Abra (ou crie) o arquivo `.gitignore` e adicione as seguintes linhas (personalize conforme necessário):

```
# Ignorar o ambiente virtual
.venv/
venv/

# Ignorar arquivo de variáveis de ambiente
.env
```

Fazendo o Primeiro Commit

1. **Staging das Alterações:**
   No VS Code, clique no ícone de "Stage All Changes" para preparar todas as alterações para commit.
2. **Mensagem do Commit:**
   Insira uma mensagem descritiva, como "Primeiro commit" e confirme o commit.
3. **Verificação:**
   Confirme que os arquivos indesejados (como `.env` e o diretório do ambiente virtual) não estão presentes na listagem de arquivos versionados, graças ao .gitignore.

Sincronizando o Repositório Local com o GitHub

- Acesse sua conta no GitHub e crie um novo repositório.
  Observação: No exemplo, o nome do repositório será "crew for Depli". Não é necessário adicionar um README ou .gitignore, pois esses arquivos já foram configurados localmente.

Adicionando o Remote no VS Code

1. **Adicionar Remote:**
   No VS Code, na área de Git, clique no ícone de "Mais ações" (três pontinhos) e escolha a opção **Remote: Add Remote**.
2. **Selecionando o Repositório:**
   O VS Code pode sugerir o repositório criado recentemente. Caso contrário, insira a URL do repositório remoto manualmente.

3. **Verificação:**
   Certifique-se de que o remote foi adicionado corretamente e que a branch principal (geralmente, *main*) está configurada para sincronização com o repositório GitHub.

Realizando o Push para o GitHub

- Com o remote configurado, clique em “Push” para enviar todas as alterações do repositório local para o GitHub.
- Verifique, na interface web do GitHub, se os arquivos apareceram corretamente e se os arquivos sensíveis (como `.env`) foram devidamente ignorados.

Conclusão e Próximos Passos

- **Revisão:**
  Aqui, você aprendeu a transformar a pasta do projeto local em um repositório Git, configurar o `.gitignore` para garantir que arquivos sensíveis ou desnecessários sejam ignorados e sincronizar o repositório com o GitHub.
- **Próximas Etapas:**
  A seguir, veremos como realizar o deploy do seu projeto utilizando o CrewAI Enterprise, que fará a conexão e a execução do deploy a partir do repositório do GitHub.

04. Fazendo deploy de uma crew

Nesta aula, vamos realizar o deploy da nossa crew utilizando o **CrewEnterprise** (a plataforma de deploy para crews). O processo é intuitivo e, embora possa levar alguns minutos, é bastante simples. Nosso objetivo é conectar o repositório hospedado no GitHub (onde já publicamos nosso projeto) à plataforma CreaEnterprise, configurar as variáveis de ambiente necessárias e lançar o deploy da nossa API.

Acessando o CrewAi Enterprise

1. **Entrando na Plataforma:**
    - Acesse o CrewAI Enterprise
    - Faça o login na plataforma.
    - Se você ainda não configurou o GitHub dentro do CrewAI Enterprise, siga as instruções passo a passo que o próprio site oferece. Essa configuração permitirá que o serviço acesse os repositórios de sua conta.
2. **Conectando o Repositório:**
    - Após configurar o GitHub, aparecerá a opção **Deploy your crews from GitHub**.
    - Selecione o repositório referente ao projeto que criamos anteriormente (no exemplo, “crew for Deploy”).
    - A plataforma reconhecerá automaticamente a branch principal (geralmente, a branch **main**).

Configurando Variáveis de Ambiente

Embora o arquivo **.env** com as variáveis não seja enviado para o repositório (por segurança), é fundamental que o deploy receba esses dados. O CreaEnterprise oferece uma interface para configurar as *environment variables* manualmente.

1. **Adicionando as Variáveis:**
    - Localize a seção de Env_Var_Keys.
    - Adicione a sua **OpenAI API key**; esse valor deve ser copiado do seu arquivo **.env** local.

- Adicione também a variável **model**, por exemplo, “gpt4omini”.
  Dica: Configure as variáveis-chave como padrão, para facilitar o processo em deploys futuros.

2. **Confirmando a Configuração:**
   - Verifique se todas as variáveis necessárias foram adicionadas corretamente.
   - Uma vez configuradas, clique em **Deploy** para iniciar o processo.

Realizando o Deploy

1. **Iniciando o Deploy:**
   - Após configurar as variáveis, clique no botão **Deploy**.
   - O processo de deploy iniciará e aparecerá um status indicando que ele está *in progress*.
   - O tempo para a conclusão do deploy pode variar: na primeira execução, o processo pode demorar até 10 minutos ou um pouco mais.
2. **Monitoramento e Logs:**
   - Durante o deploy, o painel do CreaEnterprise exibirá logs e mensagens de status.
   - Caso ocorra algum erro (por exemplo, instabilidade do servidor ou *deployment error*), há a opção de **redeploy** para reiniciar o processo sem que seja necessário modificar qualquer arquivo local ou no repositório.
3. **Finalização do Deploy:**
   - Quando o deploy for concluído com sucesso, o status deverá indicar algo como “CREWAI IS ONLINE” (ou similar).
   - Mesmo que inicialmente tenha ocorrido algum erro, ao efetuar um redeploy, o sistema normalmente completa o processo sem grandes modificações.

Considerações Finais e Próximos Passos

- **Revisão do Processo:**
  Você aprendeu como conectar o repositório do GitHub ao CrewAI Enterprise, configurar as variáveis de ambiente necessárias e realizar o deploy da sua crew.

- **Verificação da API:**
  Na próxima aula, será o momento de testar a API que foi criada. Esse teste confirmará se o deploy foi realizado com sucesso e se a aplicação está funcionando como esperado.
- **Observação Importante:**
  Se durante o processo ocorrer algum erro, utilize o recurso de redeploy fornecido pela plataforma. Geralmente, uma nova tentativa resolve problemas pontuais sem a necessidade de alterações no projeto.

05. Testando a API com Postman

Agora que você já fez o deploy da sua API, vamos aprender a testá-la utilizando o Postman. Se você ainda não conhece essa ferramenta, ela é bastante popular entre os desenvolvedores por facilitar a criação e o envio de requisições HTTP.

Postman

O Postman é uma ferramenta essencial para desenvolvedores que trabalham com APIs. Ele permite criar, enviar e monitorar requisições HTTP de forma interativa, simplificando o processo de testes e o desenvolvimento de integrações. Se você ainda não conhece o Postman, saiba que ele vai ajudá-lo a:

- **Visualizar respostas:** Exibe os resultados das requisições de forma clara, com formatação automática do JSON e outros tipos de conteúdo.
- **Automatizar testes:** Permite a criação de scripts para testar automaticamente as respostas das APIs.
- **Organizar chamadas:** Com Workspaces e coleções, você pode manter todas as suas requisições organizadas, facilitando o gerenciamento dos endpoints que compõem sua aplicação.

Caso ainda não tenha uma conta, acesse postman.com e crie a sua. Após o login, normalmente você trabalhará no “My Workspace”, que agrupa todas as suas requisições de forma prática.

Preparando a Requisição

Inicie copiando a URL da API, que foi utilizada no processo de deploy. Essa URL será a base para todas as requisições. Além disso, como a API utiliza um Bearer Token (que funciona como uma “senha”), inclua-o na seção de **Authorization** do Postman. Isso garante o acesso aos recursos protegidos da API.

Teste Inicial da API

Para confirmar que a API está operando, envie uma requisição GET simples para a URL da API. Se tudo estiver correto, a resposta indicará que a API está “Healthy” e funcionando. ### Consulta aos Parâmetros

Para saber quais parâmetros (inputs) sua aplicação requer, envie uma requisição GET para o endpoint `/inputs` (exemplo: `http://sua_url/inputs`). Essa requisição retornará uma lista com os inputs necessários, como “Topic” e “Current Year”, que deverão ser usados nas próximas chamadas.

Exemplo de Resposta

```
{
  "inputs": ["Topic", "Current Year"]
}
```

Iniciando o Processo com Kickoff

Com os inputs identificados, é hora de iniciar a execução da API utilizando o endpoint `/kickoff`. Configure uma nova requisição com método POST e insira a URL apropriada (exemplo: `http://sua_url/kickoff`). No corpo da requisição, escolha o formato raw com JSON e insira os dados conforme abaixo:

Exemplo de Corpo da Requisição

```
{
  "inputs": {
    "Topic": "CREAI",
    "Current Year": 2025
  }
}
```

Se a requisição estiver correta, a API retornará um identificador (kickoff ID) para acompanhar a execução.

Monitorando a Execução

Para acompanhar o andamento do processamento iniciado, envie uma requisição GET para o endpoint `/status/[KICKOFF_ID]`, substituindo `[KICKOFF_ID]` pelo identificador recebido. Essa requisição permitirá verificar se o processo está em execução, já concluído ou se ocorreu algum erro.

Conclusão

A utilização do Postman possibilita testar e validar todos os endpoints da sua API de maneira interativa e intuitiva. Neste capítulo, você aprendeu a:

- Configurar o Postman com a URL e o Bearer Token.
- Testar a disponibilidade da API enviando uma requisição simples.
- Consultar os parâmetros necessários (inputs).
- Iniciar a execução (kickoff) e monitorar seu status.

No próximo capítulo, veremos como realizar esses mesmos testes utilizando Python, ampliando suas ferramentas de integração.

06. Testando a API com Python

Até aqui você viu como testar sua API via Postman; agora é hora de trazer tudo para o seu ambiente Python. Neste capítulo, vamos replicar em código as mesmas chamadas de **inputs**, **kickoff** e **status** que fizemos no Postman — mas usando um *notebook* e a biblioteca `requests`. Assim, você terá exemplos prontos para incorporar aos seus projetos Python.

Preparando o ambiente

1. **Criando o notebook**
   Abra o seu editor (Jupyter, VS Code, Colab etc.) e crie um arquivo chamado `API_Testing.ipynb`.
2. **Instalando a biblioteca**
   Caso ainda não tenha, instale o `requests`:

```
pip install requests
```

3. **Defina URL e token**: no início do notebook, armazene a URL de deploy e o Bearer Token em variáveis, montando um cabeçalho de autenticação:

```
   import requests

   url = "https://crew-for-deply-7fb406fe-5558-4f49-9a53-7cf3-47c034f1.crewai.com"
   token = "90e7968cb875"
   headers = {
       "Authorization": f"Bearer {token}",
       "Content-Type": "application/json"
   }
   ```


Verificando se a API está viva

Antes de qualquer outra chamada, confirme que a API responde:

``` python
response = requests.get(url, headers=headers)
if response.status_code == 200:
   print("API está saudável:", response.text)
else:
   print(f"Erro ao conectar: {response.status_code}")
```

- **requests.get** faz o GET na URL base.
- **200** significa que a API retornou com sucesso (normalmente um JSON com `{ "status": "Healthy" }`).

Consultando os inputs

Para saber quais parâmetros a API espera, faça um GET em `/inputs`:

```
response = requests.get(f"{url}/inputs", headers=headers)
if response.status_code == 200:
    print("Inputs disponíveis:", response.json()["inputs"])
else:
    print(f"Falha ao obter inputs: {response.status_code}")
```

A saída deve listar os campos, por exemplo:

```
Inputs disponíveis: ['topic', 'current_year']
```

Iniciando uma execução (kickoff)

Com os inputs identificados, dispare a execução com um POST em `/kickoff`:

```
body = {
    'inputs': {
        'topic': 'CrewAI',
        'current_year': 2025
    }
}

response = requests.post(f"{url}/kickoff", headers=headers, json=body)
if response.status_code == 200:
    kickoff_id = response.json().get('kickoff_id')
    print("Kickoff iniciado com ID:", kickoff_id)
else:
    print(f"Erro no kickoff: {response.status_code}", response.text)
```

- `**json=body**` serializa automaticamente o dicionário em JSON.
- O retorno contém `kickoff_id`, que identifica essa execução.

Acompanhando o status da execução

Para verificar o andamento, faça um GET em `/status/{kickoff_id}`:

```
kickoff_id = '579d6d03-2b4d-4230-b716-cbcd5273615f'
response = requests.get(f"{url}/status/{kickoff_id}", headers=headers)
if response.status_code == 200:
    result = response.json().get('result')
    print("Resultado final:", result)
else:
    print(f"Erro ao checar status: {response.status_code}")
```

Quando o `state` for “success”, o campo `result` traz o output completo gerado pela sua CREW.

Conclusão

Você agora tem um notebook completo para testar a API Crew AI via Python, capaz de:

- Confirmar a disponibilidade da API;
- Listar inputs necessários;
- Iniciar execuções programaticamente;
- Acompanhar o resultado final.

Com o conteúdo apresentado, percorremos todo o fluxo de deploy de projetos na plataforma CrewAI Enterprise, da criação da API à sua validação prática com Postman e Python. A proposta foi oferecer uma base clara, aplicável e objetiva, capacitando desenvolvedores a integrarem e testarem suas soluções com autonomia e segurança.

Agora, com esses fundamentos em mãos, você está pronto para explorar novos desafios, integrar sistemas mais robustos e transformar suas ideias em aplicações inteligentes e escaláveis. A tecnologia está ao seu alcance — e seu próximo passo depende apenas da sua iniciativa.

Siga praticando, experimentando e evoluindo. O domínio do deploy com CrewAI é só o começo.