Capítulo 29 – Hackathon do Projeto Final (Cloud e Deploy)

Aléssio Miranda Jr.

Novembro - 2026/2

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

O que você verá neste capítulo

A genialidade não tem valor se não puder ser acessada. Neste capítulo, abordaremos o desafio final da engenharia de software contemporânea: o **Deploy na Nuvem** (Cloud Computing). Você aprenderá por que o jargão "na minha máquina funciona" não tem espaço no mercado, como empacotar sua aplicação de Inteligência Artificial usando conteinerização (Docker) e plataformas PaaS (*Platform as a Service*), e como garantir que suas senhas e chaves de API não caiam nas mãos de hackers durante o processo de publicação do seu projeto.

1 O Problema de "Na Minha Máquina Funciona"

Neste exato ponto do Hackathon, as equipes possuem um ecossistema de software brilhante e plenamente funcional. O modelo de inteligência artificial recupera os dados, o back-end processa a arquitetura RAG e o front-end exibe os botões em uma interface limpa.

No entanto, tudo isso está acontecendo **exclusivamente no seu laboratório** (ou no seu quarto). O servidor está operando no endereço `http://localhost:8000`. Se você copiar esse endereço e mandar no WhatsApp para o professor ou para um cliente investidor, a página retornará erro.

Nenhuma empresa real avalia um software pedindo para o cliente instalar bibliotecas no terminal. Chegamos à etapa crucial que separa amadores de profissionais: a **Nuvem** (*Cloud Computing*). O objetivo tecnológico deste en-

contro é submeter o código do projeto para a internet pública, tornando-o acessível de qualquer navegador do mundo por meio de uma URL estática (ex: `meu-projeto-ia.onrender.com`).

2 Estratégias de Hospedagem (O Ecossistema PaaS)

Historicamente, hospedar uma aplicação exigia que o desenvolvedor alugasse um servidor Linux nu (como na AWS EC2), configurasse proxies reversos (Nginx) e liberasse portas manualmente no firewall. Hoje, a cultura de DevOps desenvolveu o modelo **PaaS** (*Platform as a Service*).

Plataformas PaaS conectam-se diretamente ao seu repositório Git. Quando você envia o código, a plataforma clona o repositório, identifica a linguagem, instala os pacotes listados (no `requirements.txt` ou `pom.xml`) e roda o comando de inicialização automaticamente.

• Teoria: title=Recomendações de Hospedagem para o Hackathon

- **Para o Back-end (Python/Java):** A plataforma **Render** (render.com) e a **Railway** são excelentes portas de entrada gratuitas para APIs. Elas lidam incrivelmente bem com a compilação pesada necessária para instalar bibliotecas matemáticas do Python e expõem os serviços com certificados HTTPS automaticamente.
- **Para o Front-end (React, Vue, HTML):** O **Vercel** ou o **Netlify** são os titãs do mercado. Eles distribuem arquivos estáticos em *CDNs* globais em poucos segundos.
- **A Rota Unificada (Streamlit/Gradio):** Se a equipe optou por fundir as duas camadas em uma interface monolítica de Python puro (Streamlit), o **Streamlit Community Cloud** ou o **Hugging Face Spaces** oferecem implantação *one-click*. Eles são ecossistemas desenhados exclusivamente para IAs.

3 O Segredo e a Ameaça das Variáveis de Ambiente

A falha primária mais letal que derruba startups juniores no momento do *deploy* não é código quebrado, é a **exposição de segredos de segurança**.

No seu computador local, as senhas do banco de dados e a chave de acesso do modelo (ex: `OPENAI_API_KEY`) habitam um arquivo oculto chamado `.env`, que o git obrigatoriamente ignora (via `.gitignore`).

Tabela 1: Opções de Hospedagem em Nuvem para Startups

Plataforma	Especialidade	Vantagem no Hackathon
Render	Back-end (APIs)	Conecta direto no Git e instala bibliotecas pesadas de Python nativamente.
Vercel	Front-end (Web)	Deploy instantâneo de sites React/HTML com CDN global automática.
Hugging Face Spaces	Full-stack GenAI	Hospeda aplicações Streamlit gratuitamente com suporte a GPUs sob demanda.
Railway	Bancos de Dados	Ótimo para manter bancos SQL ou Vetoriais online persistentes.

▶ Prática: title=A Injeção de Environment Variables

Se as chaves não vão para o GitHub, como o servidor na nuvem vai se autenticar na OpenAI?
Você **jamais** cola a string da senha no seu arquivo `main.py`. Em vez disso, você acessa o painel de configurações (Dashboard) da sua plataforma na nuvem (Vercel ou Render) e procura pela aba **Environment Variables** (Variáveis de Ambiente).
Lá, você cadastra o par de "Chave = Valor" (ex: `OPENAI_API_KEY = sk-proj-...`). Durante a fase de inicialização do container, a nuvem injeta esses segredos diretamente no cérebro (memória RAM) do sistema operacional virtual, mantendo o código seguro e garantindo que, se o repositório for clonado, a chave não será roubada.

Ao dominar este fluxo (código local → git push → build na nuvem → injeção de segredos), sua startup está pronta para abrir as portas para os primeiros usuários.

✔ Takeaways

- “Na minha máquina funciona” é inaceitável: o produto só existe quando está acessível na internet via URL pública.
- Plataformas **PaaS** (Render, Vercel, Railway) automatizam o deploy: conectam-se ao Git, instalam dependências e expõem o serviço com HTTPS automático.
- **Render** é ideal para backend (Python/Java), **Vercel** para frontend (React/HTML), e **Hugging Face Spaces** para apps GenAI full-stack (Streamlit/Gradio).
- **Variáveis de Ambiente** são o mecanismo seguro para injetar chaves de API na nuvem: cadastre no dashboard da plataforma, nunca no código-fonte.
- A exposição acidental de `OPENAI_API_KEY` em um repositório público pode gerar prejuízos financeiros em minutos (bots escaneiam o GitHub em tempo real).

4 Conclusão

Com o sistema publicado na nuvem e acessível via URL pública, falta apenas o último passo: **provar o valor** do que vocês construíram. No próximo encontro, o **Demoday**, cada equipe terá 10 minutos para apresentar seu produto ao vivo para a banca — o momento culminante do semestre inteiro.

✎ Questões: Autoavaliação

1 **[Direta]** Explique o fluxo completo de deploy em uma plataforma PaaS (ex: Render): desde o `git push` até o serviço estar disponível na internet.

2 **[Direta]** Descreva o mecanismo de Environment Variables: como as chaves de API chegam ao código em produção sem estarem no repositório?

3 **[Direta]** Compare Render, Vercel e Hugging Face Spaces como opções de hospedagem. Para cada plataforma, indique o tipo de serviço mais adequado (backend, frontend, full-stack).

4 **[Reflexiva]** Uma equipe comitou acidentalmente a `OPENAI_API_KEY` no GitHub público. Descreva o plano de resposta imediata (revogar, auditar, rotacionar) e as medidas preventivas para evitar reincidência.

5 **[Reflexiva]** Discuta os trade-offs entre hospedar o Vector Store localmente (ChromaDB em memória no Render) versus como serviço gerenciado na nuvem (Pinecone). Considere custo, persistência e latência.

Lab. 29: Colocando o Produto no Ar

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

⋆ Objetivo do Laboratório

O objetivo do último dia do Hackathon é a publicação final (*Deploy*) do projeto da Startup. Ao final da aula, a aplicação não deve depender de nenhuma máquina local do laboratório para responder requisições, estando totalmente migrada para plataformas de Nuvem.

5 Atividade Prática

1. Preparação para Produção

1. Revise o `requirements.txt` ou o `pom.xml`. Eles devem conter as versões exatas de todas as bibliotecas usadas pela equipe, para que o servidor consiga espelhar a máquina de vocês perfeitamente.
2. Certifique-se de que o projeto principal roda automaticamente a partir de um comando base sem depender de caminhos absolutos locais (como `C:/Users/aluno/banco.pdf`). Utilizem caminhos relativos.

2. Deploy do Banco de Dados e Vetores (Opcional, se aplicável)

Se a aplicação utilizar um *Vector Store* atrelado à memória, saibam que algumas plataformas *Serverless* limpam a memória ao reiniciar o contêiner. Se os documentos foram *chunkados* corretamente, este será o momento para subir a coleção de PDFs e rodar o script de pré-ingestão uma única vez na nuvem.

3. Deploy das Aplicações

- **Criem a Conta Cloud:** Escolham o provedor (Render, Vercel, Railway, Streamlit Share, etc).

- **Conectem ao Git:** O provedor deve buscar o código nativamente através de uma permissão OAuth com o GitLab.
- **Ajustem as Variáveis:** Insiram todas as chaves de segurança (`API_KEYs`) no painel de *Environment Variables* do serviço escolhido.
- **Desencadeiem o Build:** Acompanhem as abas de *Logs*. A maioria das falhas de compilação acontece devido a versões incompatíveis do Python/Java.

4. Preparação para o Pitch

O fim desta atividade marca a finalização do escopo prático da disciplina. A equipe deve elaborar a defesa do produto (Pitch), montando a justificativa técnica das tecnologias selecionadas em preparação para a aula do **Demoday**.

6 Critério de Avaliação

O laboratório e o Hackathon estão concluídos quando o Produto Funcional estiver comprovadamente publicado. O professor testará a aplicação através do link gerado e constatará o encerramento das *Issues* no GitLab da disciplina.

✔ Takeaways

- O produto está **publicado e acessível na internet** — qualquer pessoa com a URL pode usá-lo, sem depender da máquina do laboratório.
- As chaves de API foram injetadas via **Environment Variables** no painel da plataforma — nenhum segredo no código.
- O `requirements.txt` / `pom.xml` com versões fixas garante que o build na nuvem replica exatamente o ambiente local.
- O Pitch está esboçado: Dor → Solução → Demo → Engenharia — o roteiro de 10 minutos para o Demoday.

◇ Informação

Gancho para o Demoday: O produto está no ar. Amanhã é o momento de **provar o valor** do que vocês construíram. Cada equipe terá 10 minutos para apresentar ao vivo, responder perguntas técnicas e defender as decisões arquiteturais. Ensaiem o Pitch: quem fala o quê, em que ordem, e pratiquem a demo ao vivo na URL pública.