In [1]:
```
import os
import requests
from dotenv import load_dotenv
from bs4 import BeautifulSoup
from IPython.display import Markdown, display
```

In [2]:
```
class Website:

    def __init__(self, url):
        """
        Create this Website object from the given url using the Beautiful
        """
        self.url = url
        headers = {
         "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWe
        }
        response = requests.get(url, headers=headers)
        soup = BeautifulSoup(response.content, 'html.parser')
        self.title = soup.title.string if soup.title else "No title found
        for irrelevant in soup.body(["script", "style", "img", "input"]):
            irrelevant.decompose()
        self.text = soup.body.get_text(separator="\n", strip=True)
```

In [3]:
```
system_prompt = "You are an assistant that analyzes the contents of a web
and provides a short summary, ignoring text that might be navigation rela
Respond in markdown."
```

In [4]:
```
def user_prompt_for(website):
    user_prompt = f"You are looking at a website titled {website.title}"
    user_prompt += "\nThe contents of this website is as follows; \
please provide a short summary of this website in markdown. \
If it includes news or announcements, then summarize these too.\n\n"
    user_prompt += website.text
    return user_prompt
```

In [5]:
```
def messages_for(website):
    return [
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": user_prompt_for(website)}
    ]
```

In [6]:
```
OLLAMA_API = "http://localhost:11434/api/chat"
HEADERS = {"Content-Type": "application/json"}
MODEL = "llama3.2"
```

In [7]:
```
def summarize(url):
    website = Website(url)
    payload = {
        "model": MODEL,
        "messages": messages_for(website),
        "stream": False
    }
    response = requests.post(OLLAMA_API, json=payload, headers=HEADERS)
    return response.json()['message']['content']
```

In [8]:
```
def display_summary(url):
    summary = summarize(url)
    display(Markdown(summary))
```

In [9]:
```
display_summary("https://www.infoworld.com/article/3893387/how-terraform-
```

The article discusses the current state of Infrastructure as Code (IaC) market, specifically focusing on HashiCorp's Terraform. Here are the main points:

1. **Terraform remains dominant**: Despite facing increased competition from open-source alternatives like OpenTofu and Pulumi, Terraform remains the leading force in the IaC market.
2. **Evolution to support complex enterprise requirements**: HashiCorp has adapted its product to cater to more complex enterprise needs, making it a more effective solution for larger organizations.
3. **Strategic emphasis on infrastructure life cycle**: By focusing on the entire infrastructure life cycle, Terraform positions itself favorably to address evolving cloud-era needs.
4. **Continuous innovation and security commitment**: HashiCorp's ongoing investment in product development and security ensures that Terraform stays ahead of the competition.
5. **Vibrant partner ecosystem**: The robust partner ecosystem supports multicloud compatibility, making it easier for customers to adopt Terraform across various cloud providers.

The article concludes that while OpenTofu, Pulumi, and other open-source options are a threat, HashiCorp's strategic focus on enterprise requirements, innovation, and security ensures its position as the dominant IaC solution.

Key Takeaways:

- Terraform's dominance in the IaC market is unlikely to be challenged by open-source alternatives.
- HashiCorp's evolution of Terraform addresses complex enterprise needs, making it a more effective solution.
- The company's focus on infrastructure life cycle and security ensures its competitive advantage.
- Continuous innovation and investment in security are key to maintaining Terraform's position as the leading IaC solution.

In [ ]:
```
```