Technology Explained

A Simple Guide to Understanding

the Tech Stack

Breaking down complex technologies into everyday language

Technologies Covered:

• Python and FastAPI

• .NET and ASP.NET Core

• React and Frontend Development

• Node.js and API Gateways

• Azure OpenAI and GPT-4

• Databases (SQL, NoSQL, Vector)

• Docker and Containers

• Kubernetes

• CI/CD and GitHub Actions

• Cloud Platforms (Azure/AWS)

1. Python and FastAPI

What is Python?

Python is a programming language that's known for being easy to read and write. Think of it like English compared to other programming languages that might be more like Latin - Python uses words that make sense and doesn't require lots of complicated symbols.

> **Real-World Analogy:** If programming languages were cars, Python would be like a Tesla - modern, user-friendly, and powerful. It has lots of pre-built features (libraries) that let you do complex things without building everything from scratch.

Why use Python for AI?

Python has become the go-to language for AI and data science because it has tons of ready-made tools for working with data, connecting to AI services, and processing information. It's like having a toolbox where someone already invented all the tools you need - you just use them.

What is FastAPI?

FastAPI is a framework for building web APIs (Application Programming Interfaces). An API is like a waiter in a restaurant - you tell the waiter what you want, the waiter goes to the kitchen and gets it, then brings it back to you. FastAPI makes it easy to create these 'waiters' that handle requests from websites or apps.

> **Simple Example:** When you ask the AI assistant a question on a website, your question goes to a FastAPI server. The server processes your question, gets an answer from the AI, and sends it back to display on your screen. All of this happens in seconds.

Why FastAPI specifically?

FastAPI is 'fast' in two ways. First, it runs quickly - it can handle many requests at once without slowing down. Second, it's fast to write code with - it automatically creates documentation and catches errors before they become problems.

How it's used in the project:

In this project, FastAPI runs the AI service. When you ask a question to the AI assistant, FastAPI receives your question, searches through documents to find relevant information, sends everything to GPT-4, gets the answer, and sends it back to you. It does all this while handling requests from many users at the same time.

2. .NET and ASP.NET Core

What is .NET?

.NET (pronounced 'dot net') is a platform created by Microsoft for building applications. Think of it as a foundation and toolset for building software. It's been around for over 20 years and is trusted by many large companies for building serious business applications.

> **Real-World Analogy:** If Python is like a Tesla, .NET is like a commercial-grade truck - it's built for reliability, can handle heavy loads, and has been tested in all conditions. Big companies love it because it's stable and well-supported.

What is ASP.NET Core?

ASP.NET Core is a specific part of .NET designed for building web applications and services. The 'Core' part means it's the modern version that works on any computer (Windows, Mac, Linux), not just Windows computers like older versions.

What is C#?

C# (pronounced 'C sharp') is the programming language used with .NET. It's similar to other languages like Java. C# is very structured and strict, which helps prevent mistakes. It's like having a very detail-oriented assistant who double-checks everything you write.

What is Entity Framework?

Entity Framework is a tool that helps .NET applications work with databases. Instead of writing complicated database commands, you can work with data like regular objects in your code.

> **Simple Example:** Imagine you have a filing cabinet (database) full of folders (data). Without Entity Framework, you'd need to know exactly which drawer, which folder, and describe exactly what you want. With Entity Framework, you just say 'get me all customer records' and it figures out the rest.

How it's used in the project:

The .NET service handles all the governance work - tracking AI projects, calculating risk scores, checking compliance with company rules, and keeping audit records. It's perfect for this because governance data is structured (organized in specific ways) and needs to be handled carefully with no mistakes.

For example, when someone registers a new AI project, the .NET service saves all the details to the database, calculates how risky the project is, checks if it follows company policies, and records who approved it and when.

3. React and Frontend Development

What is the Frontend?

The frontend is everything you see and click on in a website or app. It's the buttons, forms, text, images, and animations. Think of a restaurant - the frontend is the dining room where customers sit, while the backend is the kitchen where food is prepared.

What is React?

React is a JavaScript library for building user interfaces. It was created by Facebook and is now used by millions of websites including Netflix, Instagram, Airbnb, and many others.

> **Real-World Analogy:** Think of React like LEGO blocks. Instead of building an entire house from scratch, you create reusable blocks (called components). A button is one component, a form is another. You can use these blocks over and over in different places. If you want to change all buttons, you just change the button component once.

What are Components?

Components are the building blocks of a React application. Each component is a piece of the user interface that can be reused. For example, you might have a 'Search Box' component, a 'User Profile' component, and a 'Navigation Menu' component.

The genius of components is that they're independent. You can update one without breaking others. It's like being able to replace the steering wheel in a car without affecting the engine.

What is Vite?

Vite (pronounced 'veet', French for 'fast') is a tool that helps developers build React apps quickly. When you're developing, it shows changes instantly - you save your code and see the results on the screen in less than a second. This makes development much faster.

What is TailwindCSS?

TailwindCSS is a tool for styling websites (making them look good). Instead of writing custom style code for every element, Tailwind provides pre-made classes like 'text-blue' or 'rounded-button'. It's like having a paint-by-numbers kit instead of starting with a blank canvas.

How it's used in the project:

React powers the entire user interface. The dashboard you see, the AI chat window where you ask questions, the project registry, the risk assessment screens - all of these are React components. The

interface is responsive, meaning it works well on phones, tablets, and computers.

When you type a question to the AI, React handles showing your message, displaying a loading indicator while waiting, and then showing the AI's response with proper formatting. All of this happens smoothly without the page needing to reload.

4. Node.js and API Gateway

What is Node.js?

Node.js lets you run JavaScript on a server, not just in web browsers. JavaScript used to only work in browsers, but Node.js changed that. Now you can use the same language for both the frontend (what users see) and backend (server logic).

> **Real-World Analogy:** Imagine if you learned Spanish to talk to customers, and then found out you could also use Spanish to talk to your coworkers. That's Node.js - one language for everything makes development easier.

Why is Node.js good for servers?

Node.js is 'non-blocking', which is a fancy way of saying it can handle many things at once efficiently. Think of a single chef who starts cooking multiple dishes at the same time - while one dish is in the oven, they prep another one. Node.js works the same way with requests.

What is Express?

Express is a framework for Node.js that makes building web servers easier. It provides structure and shortcuts for common tasks like handling different types of requests, managing cookies, and sending responses. Without Express, you'd have to write a lot more code from scratch.

What is an API Gateway?

An API Gateway is like a receptionist at a big office building. Instead of visitors wandering around trying to find the right office, they check in with the receptionist first. The receptionist verifies who they are, figures out where they need to go, and directs them to the right place.

> **In Technical Terms:** When your browser sends a request, it goes to the API Gateway first. The gateway checks if you're logged in, decides which backend service should handle your request (Python AI service or .NET governance service), sends the request there, gets the response, and sends it back to you.

What does Authentication mean?

Authentication is proving who you are. It's like showing your ID at an airport. The API Gateway checks your login token (a special encrypted code) to make sure you are who you claim to be before letting your request through.

What is Rate Limiting?

Rate limiting prevents abuse by limiting how many requests one person can make in a time period. It's like a buffet restaurant that says 'you can come back for seconds, but only after 5 minutes'. This stops people from overwhelming the system with too many requests.

5. Azure OpenAI and GPT-4

What is Azure?

Azure is Microsoft's cloud platform. Instead of buying and maintaining your own computers and servers, you rent computing power from Microsoft. It's like renting an apartment instead of buying a house - you get what you need without the maintenance headaches.

What is OpenAI?

OpenAI is the company that created ChatGPT and GPT-4. They make some of the most advanced AI models in the world. These models can understand and generate human-like text, answer questions, write code, and much more.

What is Azure OpenAI?

Azure OpenAI is OpenAI's technology available through Microsoft's Azure cloud. For businesses, this is important because it means the AI runs in a secure, enterprise-grade environment with proper data protection and compliance features. Your company data doesn't mix with other companies' data.

What is GPT-4?

GPT-4 is one of the most advanced AI language models. It has been trained on massive amounts of text from the internet, books, and other sources. This training helps it understand context, answer questions, generate text, and even reason through problems.

> **How it works - Simple Version:** Imagine someone who has read millions of books and articles. When you ask them a question, they can draw on all that knowledge to give you a helpful answer. That's essentially what GPT-4 does, except it's software, not a person.

What are Embeddings?

Embeddings are a way to convert text into numbers that capture meaning. Words or sentences that mean similar things get similar numbers. This lets computers understand that 'car' and 'automobile' are related, or that 'happy' is more similar to 'joyful' than to 'sad'.

> **Real-World Example:** Think of it like a map coordinate system. Cities that are close together have similar coordinates. Words with similar meanings have similar embedding numbers. This lets us search for meaning, not just exact word matches.

How it's used in the project:

Azure OpenAI provides two main services for this project. First, GPT-4 powers the AI assistant that answers user questions. Second, the embedding model converts all company documents into embeddings so we can search them by meaning, not just keywords. When you ask a question, the system finds relevant documents using embeddings, then sends those documents and your question to GPT-4 to generate an accurate answer.

6. Databases (SQL, NoSQL, Vector)

What is a Database?

A database is an organized collection of data, like a very smart filing system. Instead of scattered papers in drawers, everything is organized so you can find and update information quickly. Every app and website uses databases to store information.

SQL Server - Relational Database

SQL (Structured Query Language) databases store data in tables with rows and columns, like Excel spreadsheets. They're great when data has a clear structure. SQL Server is Microsoft's version of this type of database.

> **Real-World Analogy:** Think of a library catalog system. Each book has specific information: title, author, ISBN, publish date. This structured information fits perfectly in a SQL database. You can easily find 'all books by this author' or 'all books published after 2020'.

Used in the project for: Storing governance data like AI projects, risk assessments, approval records, and audit logs. This data is structured and needs to be reliable - if a project is approved, that record must never be lost or corrupted.

Cosmos DB - NoSQL Database

NoSQL databases store data in a flexible format, like JSON documents. They don't require a fixed structure - different records can have different fields. Cosmos DB is Microsoft's globally distributed NoSQL database.

> **Real-World Analogy:** Think of a scrapbook versus a form. A form (SQL) requires specific fields filled out the same way every time. A scrapbook (NoSQL) lets you add whatever you want - photos, notes, tickets - in any format. More flexible, but less structured.

Used in the project for: Storing AI conversation history and metadata. Each conversation might have different types of information - some with images, some with long context, some with tool calls. NoSQL handles this flexibility well.

Pinecone - Vector Database

A vector database stores those embedding numbers we talked about earlier. It's specialized for finding similar items based on meaning, not exact matches. Regular databases can't do this efficiently.

> **Real-World Analogy:** Imagine a music recommendation system. It doesn't just match exact song titles - it finds songs that sound similar or have similar moods. Vector databases do this for text, finding documents that are semantically similar to your question.

Used in the project for: Storing embeddings of all company documents. When you ask a question, the vector database quickly finds the most relevant documents, even if they don't contain the exact words you used.

7. Docker and Containers

What is a Container?

A container is a package that includes an application and everything it needs to run - the code, settings, system tools, and libraries. It's completely self-contained and isolated from other containers.

> **Real-World Analogy:** Think of shipping containers on cargo ships. You can put anything inside - furniture, electronics, food. The container protects what's inside and makes it easy to transport. You can stack them, move them, and they work the same way anywhere. Software containers work the same way for applications.

What is Docker?

Docker is the most popular tool for creating and running containers. It packages your application so it runs exactly the same way on any computer - your laptop, a test server, or production servers. No more 'but it worked on my machine!' problems.

What is a Dockerfile?

A Dockerfile is a recipe for building a container. It lists all the ingredients (software dependencies) and instructions (setup steps) needed to create the container. Anyone can use the same Dockerfile to build an identical container.

Example instructions in a Dockerfile: Start with Python 3.11, install FastAPI and other libraries, copy the application code, expose port 8000, run the application. Docker follows these steps to create the container.

What is Docker Compose?

Docker Compose lets you run multiple containers together. Most applications need several services - a web server, a database, maybe a cache. Docker Compose starts them all with one command and makes sure they can talk to each other.

> **Simple Example:** Our project has 4 services (Python AI, .NET Governance, React Frontend, Node.js Gateway). Instead of starting each one manually, Docker Compose starts all 4 with 'docker-compose up'. It's like pressing one button to turn on all the lights in your house instead of flipping each switch individually.

Why use containers?

Containers solve many problems. They ensure consistency - the app runs the same everywhere. They provide isolation - one app can't mess up another. They're lightweight - you can run many containers on one computer. They're portable - move them between different cloud providers easily.

8. Kubernetes and Orchestration

What is Kubernetes?

Kubernetes (often shortened to K8s) is a system for managing containers in production. While Docker creates and runs individual containers, Kubernetes manages thousands of containers across many computers, making sure everything runs smoothly.

> **Real-World Analogy:** Think of a large restaurant chain. Docker is like having a good recipe for cooking food. Kubernetes is like having a management system that handles all the restaurants - hiring staff when busy, closing locations that aren't needed, making sure ingredients are stocked, redirecting customers if one location is full. It manages the whole operation automatically.

What problems does Kubernetes solve?

1. Scaling: If your AI service suddenly gets 10 times more traffic, Kubernetes automatically starts more copies of that service. When traffic drops, it scales back down. You don't pay for resources you're not using.

2. Reliability: If a container crashes, Kubernetes immediately starts a new one. If a whole computer fails, Kubernetes moves all its containers to healthy computers. Users never notice the problem.

3. Load Balancing: When you have multiple copies of a service running, Kubernetes distributes incoming requests evenly across all of them. No single container gets overwhelmed while others sit idle.

4. Rolling Updates: When you need to update your application, Kubernetes gradually replaces old containers with new ones. Some old ones keep running while new ones start, so there's no downtime. Users don't even notice the update happening.

Key Kubernetes Concepts:

Pods: The smallest unit in Kubernetes. Usually contains one container, but can have multiple containers that work closely together. Think of it as a small team that works on one task.

Deployments: Describes how many copies of your application should run. 'I want 3 copies of the AI service running at all times.' Kubernetes maintains this desired state automatically.

Services: Provides a stable address for reaching your application. Even though containers come and go, the service address stays the same. It's like a permanent phone number that redirects to wherever you currently are.

ConfigMaps and Secrets: Store configuration and sensitive data separately from your application code. You can change settings without rebuilding containers. Secrets are encrypted to protect sensitive information like passwords.

9. CI/CD and DevOps

What is CI/CD?

CI/CD stands for Continuous Integration and Continuous Deployment. It's a way of automatically testing and deploying code changes. Instead of manually running tests and deploying updates, everything happens automatically when developers push code changes.

> **Real-World Analogy:** Imagine a car factory with an automated assembly line. Every part is checked as it's installed, and defects are caught immediately. The car moves through testing automatically, and only perfect cars make it to the showroom. CI/CD does this for software.

What is GitHub Actions?

GitHub Actions is a tool that automates tasks when you push code to GitHub. You define a workflow - a series of steps to run automatically. For example: run tests, build Docker images, deploy to production. GitHub Actions executes these steps every time you push new code.

How the CI/CD Pipeline Works:

Step 1 - Code Push: A developer writes new code and pushes it to GitHub. This automatically triggers the CI/CD pipeline.

Step 2 - Automated Testing: GitHub Actions runs all tests automatically. Unit tests check individual functions, integration tests check that services work together, and end-to-end tests simulate real user interactions.

Step 3 - Build Docker Images: If all tests pass, the system builds Docker images for each service. These images are tagged with version numbers and pushed to a container registry.

Step 4 - Deploy to Cloud: The new Docker images are deployed to Kubernetes in the cloud. Kubernetes performs a rolling update, gradually replacing old containers with new ones.

Step 5 - Verify: Health checks confirm the new version is working correctly. If something goes wrong, Kubernetes automatically rolls back to the previous version.

What is DevOps?

DevOps is a culture and set of practices that brings together development (writing code) and operations (running systems). Instead of developers throwing code 'over the wall' to operations teams, everyone works together with shared tools and responsibilities.

Benefits of CI/CD:

Faster deployment - updates go from code to production in minutes instead of weeks. Fewer bugs - automated testing catches problems before users see them. More confidence - if tests pass, you know the code works. Easy rollback - if something breaks, you can quickly return to the previous version. Better collaboration - everyone sees the same test results and deployment status.

10. Cloud Platforms (Azure & AWS)

What is Cloud Computing?

Cloud computing means using computers and services over the internet instead of owning physical hardware. Instead of buying servers, installing them in your office, maintaining them, and paying for electricity and cooling, you rent computing power from companies like Microsoft or Amazon.

> **Real-World Analogy:** It's like the difference between owning a car versus using Uber. With a car, you pay upfront, maintain it, insure it, and it sits unused most of the day. With Uber, you only pay when you need a ride. Cloud computing is the 'Uber' model for computers.

What is Azure?

Azure is Microsoft's cloud platform. It offers hundreds of services - virtual computers, databases, AI tools, storage, networking, and more. Companies like Walmart, Adobe, and BMW use Azure to run their applications.

What is AWS?

AWS (Amazon Web Services) is Amazon's cloud platform and is the largest cloud provider. It powers Netflix, Airbnb, NASA, and millions of other websites and applications. AWS was the first major cloud platform and has the most services available.

Key Cloud Services Used:

Azure OpenAI / AWS Bedrock: Managed AI services that provide access to powerful language models like GPT-4. You don't need to know how to train AI models - just use the API and the cloud provider handles all the complexity.

Azure Cosmos DB / AWS DynamoDB: Globally distributed NoSQL databases. Your data is automatically copied to multiple data centers around the world, so users get fast access no matter where they are, and your data is safe even if a whole data center fails.

Azure Container Apps / AWS ECS: Services that run your Docker containers in the cloud. You give them your container images, and they handle running them, scaling them, and keeping them healthy. You don't manage servers - just your application code.

Azure Key Vault / AWS Secrets Manager: Secure storage for sensitive information like passwords, API keys, and certificates. These services encrypt your secrets and control who can access them. Much safer than storing passwords in code or configuration files.

Why Use Multiple Cloud Providers?

The project is designed to work with both Azure and AWS. This avoids vendor lock-in - you're not stuck with one provider. It also gives you negotiating power on pricing and lets you use the best service from each provider. If one provider has an outage, you could potentially switch to the other.

Benefits of Cloud:

Pay only for what you use, scale up or down instantly, automatic backups and disaster recovery, global reach with data centers worldwide, access to cutting-edge AI and other services, no hardware maintenance, built-in security features, and automatic updates.