CHAPTER 1

Introduction to AI Solutions on Microsoft Azure

The AI gold rush is here—but instead of pickaxes, everyone's scrambling for GPUs and prompt engineering skills. Last quarter, a retail client told me their board demanded "generative AI something" by Friday...but their team was still manually tagging product images. That's the chaos driving the AI-102 certification's surge: companies aren't just chasing ChatGPT headlines—they're desperate for engineers who can turn Azure's toolbox into actual business wins.

Therefore, there's no better time than now to take the AI-102 exam and get certified for your skills in working with AI solutions on Microsoft Azure. It'll help boost your credibility as an AI engineer and set you apart in this job market, where there's huge demand for personnel who possess AI skills.

In this chapter, we'll examine the current AI landscape and explore what will be expected from you as an AI-102 exam candidate. I'll also help you prepare your Microsoft Azure environment for usage in developing solutions and gain an understanding of Azure AI's capabilities.

I recommend that you complete the AI-900 certification before beginning this study guide (though it's not mandatory for you to do so). This guide aims to walk you through getting certified from start to finish, and having exposure to common Microsoft Azure AI concepts will be helpful as you work through it.

Introduction to the AI Landscape

Picture this: it's 1956, and a handful of scientists at a Dartmouth College workshop[1] are huddled around punch cards, dreaming of machines that "think." Back then, building AI was like writing out every step of a recipe for someone who's never cooked before, from how to crack an egg to when to stir, leaving no room for improvisation. Fast-forward to the 1990s, and machine learning flipped the script. Suddenly, instead of handcoding how to spot a cat in a photo, we let algorithms binge-watch thousands of images until they figured it out themselves. And now, in the 2020s, working with Azure AI feels less like babysitting those early rulebooks and more like collaborating with a partner who's read every manual ever written and somehow stayed awake through it all.

What was Microsoft's big play? Consolidating its AI tools under one roof with Azure AI Foundry. Think of it as your AI workshop. Need a quick text translator? Grab the prebuilt Azure AI Translator off the shelf. Want to craft a custom chatbot that sounds like your CEO? Fire up GPT-4 in Azure OpenAI in the same workspace. But here's the catch I've seen trip up teams: choosing between these tools isn't about "advanced versus basic"—it's like choosing between a power drill and a Swiss Army knife.

Let's say you're building a medical app. A fine-tuned Azure AI Vision API could analyze X-rays out of the box, while Azure OpenAI's models could generate patient summaries that even your time-crunched nurses would trust. What the docs won't tell you is that fine-tuned vision API might cost three times more per scan than a production-ready model—a trade-off that keeps CFOs up at night. Yet when rare conditions require detection of subtle image markers, or when regulatory compliance demands explainable, domain-specific insights, it can be worth the extra expense to fine-tune a model so patients receive more accurate diagnoses and hospitals avoid costly errors.

Additionally, while representational state transfer (REST) APIs remain the primary interface for many Azure AI offerings, developers may benefit from an understanding of asynchronous (async) programming patterns, such as those they may use in async/await in Python when trying to handle long-running AI operations successfully. Familiarity with Azure fundamentals such as resource groups, networking, and cost management will also be increasingly critical for developers as AI solutions scale into enterprise settings.

In this section, I'll introduce what is expected from you as a candidate taking the AI-102 exam, where we are now with AI, and the fundamental concepts you'll need

1 John McCarthy, "The Dartmouth Workshop—As Planned and as It Happened" (*https://oreil.ly/QruL3*), October 30, 2006.

to know to understand AI's usage within the Microsoft Azure environment for the AI-102 certification.

The Minimally Qualified Candidate for the AI-102 Exam

Think of the AI-102 as your "commercial driver's license" for Azure AI—it's where you prove you can haul real-world AI solutions, not just cruise around with theory. It builds on the AI-900, which is the foundational certification test that focuses on fundamental machine learning and AI concepts that are within the Microsoft Azure ecosystem. Candidates are expected to be proficient in the different phases of AI solution development, which include requirements definition, design, development, deployment, integration, maintenance, and monitoring. This makes the AI-102 a highly interdisciplinary exam that requires you not only to understand the development process of AI systems but also to be familiar with other aspects of the system lifecycle.

To prepare yourself to take the exam, you will need to gain an understanding of the services that make up the Azure AI portfolio and know when to apply which service to which situation. This will include gaining an understanding of the data sources that you will be working with for the AI services you will utilize. You must also abide by the principles of responsible AI that Microsoft has established, be able to use REST APIs and software development kits (SDKs) to consume the Azure AI services, and have the REST APIs and SDKs integrated with applications in your own environments. We will examine this further later in the chapter.

Candidates also need to be comfortable with at least one language that's used to work with AI solutions. Throughout this book, we will be using Python for developing our AI solutions. It's a popular choice among AI engineers due to its extensive library ecosystem—it has SDKs available for many different AI services from major cloud providers and third parties. Additionally, its simplicity and readability make it an easy language to learn, write, and understand, which are essential characteristics for any language that's used to write the complex algorithms used in AI. While Python is widely used, other languages, such as R (which excels at statistical modeling and data visualization) and Java (which offers robust libraries for production-scale AI systems), also play important roles in AI development.

If you don't have much exposure to Python, I recommend that you consult some beginner-level documentation at the official Python website (*https://www.python.org*), take some beginner-level tutorials there, and learn a bit more about the use cases of the language to help you understand the foundational syntax we will be using throughout this guide.

This guide is meant not only to help you get through the exam, but also to provide you with valuable skills that you can apply to your work as an AI engineer. It will help

you demonstrate your skillset practically in the industry, where there's current high demand for AI skills.

Candidates for the AI-102 exam should also be prepared to understand and configure role-based access control (RBAC) within Azure. For instance, Azure provides built-in roles such as "Cognitive Services Contributor" and "Cognitive Services Reader." You may also create custom roles to manage access more precisely. For example, an enterprise might allow only certain data scientists to deploy models, while giving a broader group of analysts permission to run or test those models. Understanding how these permissions work in typical enterprise scenarios, such as controlled development environments and production deployments, will help you ensure that your AI solutions remain both secure and compliant.

Now that you have an understanding of what you need for the exam, we can move on to taking a look at the current state of AI.

Where We Are Now with AI

Today, AI is a rapidly growing field that has applications in almost every industry. From healthcare to finance, we can see AI being used in automating tasks, making predictions, and improving day-to-day decision making. To an extent, it has become advanced enough that it has surpassed human performance in certain areas, such as image recognition and natural language processing (NLP).

Given expectations that AI will contribute $15.7 trillion to the global economy by 2030,[2] it's no surprise that the demand for AI professionals is continuously rising, with organizations looking for skilled individuals to implement AI solutions. Various providers offer these AI technologies, whether in the form of specific solutions or centralized platforms hosted in the cloud. Using platforms within the cloud has made it easier and more affordable for many to leverage the advantages of AI. Previously, training your own model could be very expensive for the average organization. The rise of cloud-based AI platforms such as Amazon Web Services, Microsoft Azure, and Google Cloud has eliminated the need for users to invest in expensive hardware and infrastructure.

One of the leading fields in AI is *generative AI*, which is artificial intelligence that can create new and original content, including images, video, music, and text. It has many potential applications, such as creating realistic marketing campaign plans, virtual environments for video games, and new forms of art and music. We will discuss how to leverage the full potential of generative AI in Microsoft Azure in Chapter 8 of this guide.

[2] International Data Corporation, "IDC: Artificial Intelligence Will Contribute $19.9 Trillion to the Global Economy Through 2030 and Drive 3.5% of Global GDP in 2030" (*https://oreil.ly/xrnZB*), September 17, 2024

In recent developments, Microsoft has consolidated many of its cognitive services under a single Azure AI umbrella, thus simplifying how developers select and integrate AI services. Also, the Azure OpenAI Service has emerged as a key offering for organizations that want to harness large foundation models (including those based on GPT) to create advanced conversational agents, content generation systems, and more.

According to various market analyses, Azure has solidified a notable position in enterprise AI adoption, particularly as companies seek to integrate prebuilt AI capabilities. Microsoft's continued investment in GPT-based services (including ChatGPT) underscores the growing importance of foundation models for enterprise use cases, from text summarization to code generation.

Fundamental AI Concepts

There are several fundamental AI concepts that you need to be familiar with. If you have completed the AI-900 exam, you most likely know these concepts already. They include machine learning, deep learning, natural language processing, computer vision, and cognitive services. We will discuss each of these capabilities in "Gaining an Understanding of Azure AI's Capabilities" on page 18.

First off, it's important to distinguish between four interrelated disciplines (see Figure 1-1): data science, machine learning, artificial intelligence, and deep learning.

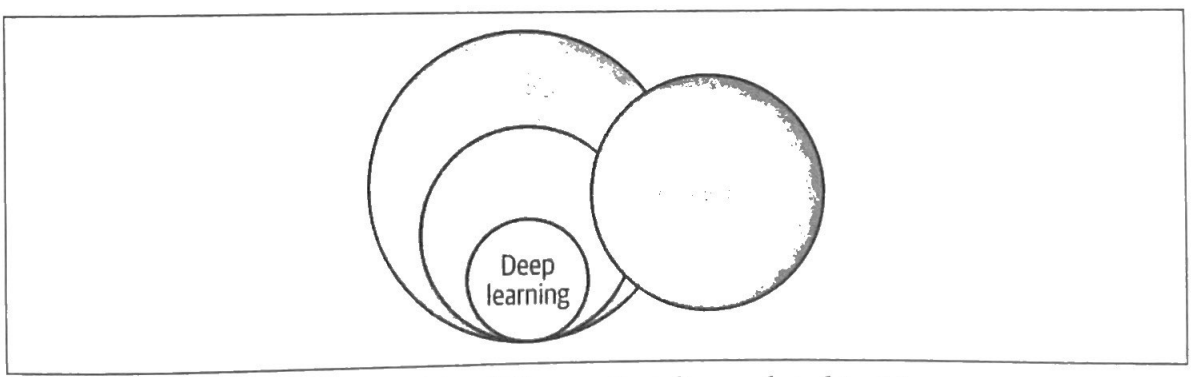

Figure 1-1. The intersections of the different disciplines related to AI

Data science

Data science is an interdisciplinary field that combines mathematics, statistics, programming, and domain expertise to analyze and interpret complex data. It uses processes and algorithms to extract knowledge and insights from both structured and unstructured data. Data science is focused on making sense of data, finding patterns, and making predictions, so it involves a lot of data processing and analysis. AI is one of the tools used in data science. Its applications go beyond data analysis and include the creation of systems that mimic human intelligence.

Machine learning

Machine learning is a type of AI that allows computers to learn from data without the need for human programmers to explicitly program them. It's used in a few Azure AI services, such as Azure Machine Learning (Azure ML). Machine learning focuses on training and validating predictive models that identify relationships between the features within the data. For instance, it can be used to create a generalized model that predicts which parts of a country are more likely to experience crime, and hence where to position more security resources.

Machine learning can be categorized into two types: supervised and unsupervised. *Supervised learning* is performed by a function that maps an input to an output based on input/output pairs. This function is inferred from labeled training data, which consists of a set of training examples. A supervised learning algorithm uses these pairs to produce the inferred function, which is then used to map new examples. Some examples of this are algorithms that detect spam in emails, perform sentiment analysis, and recognize images.

On the other hand, *unsupervised learning* is performed by training an algorithm on data without providing specific instructions on what to do with it. This allows the system to try to learn the patterns and the structure from the data without the given labels, and it encourages the algorithm to find underlying patterns within the data. Some popular use cases for unsupervised learning are performing customer segmentation and anomaly detection. *Customer segmentation* is the process of dividing customers into groups based on shared characteristics to tailor marketing strategies and product offerings to them. *Anomaly detection* is the identification of unusual patterns or outliers in data that do not conform to expected behavior, which is often used for fraud detection or system health monitoring.

Both supervised and unsupervised learning can be further divided into different types of tasks. The most popular types of tasks performed in supervised machine learning are regression and classification.

Regression problems address targets that lie on an ordered or continuous scale. This can be actual numeric values (salary, weight) or labels that have a clear rank such as "low / medium / high risk." These problems try to model the relationship between the input features and the output variable by using a mapping function, thus allowing the prediction of numerical or ordinal values based on previous observations of data. A couple of common regression-problem algorithms include linear regression and polynomial regression, which are statistical methods used to model and analyze the relationships between a dependent variable and one or more independent variables. Linear regression assumes a straight-line relationship, while polynomial regression accommodates more complex, curved relationships.

Classification problems, in contrast, predict membership in categories that lack a natural ordering (nominal classes); for example, email versus non-email, or serotypes A,

B, and C of a virus. These problems can be split into two types of classification: binary and multiclass. Binary classification problems are the simplest kind of classification problems, in which there are only two classes to predict. The email-spam flagging example is one use case of this. In contrast, multiclass classification involves prediction of multiple classes, in which each data point is classified into one of multiple possible classes. For instance, a movie-genre classification algorithm can classify a movie into one of many different genres.

On the unsupervised learning side, a fundamental task is *clustering*, which aims to group a set of objects in such a way that the objects in the same group are more similar to each other than to those in other groups. There are no labels to guide the grouping in unsupervised learning, so the algorithm is left to uncover the structure within the data and find underlying patterns on its own. A common clustering algorithm is *K-means*, which partitions data into *K* distinct, nonoverlapping subsets that are chosen to minimize within-cluster variance, based on the distance from each point to the cluster centroid. Another approach is *hierarchical clustering*, which builds a hierarchy of clusters using either a bottom-up or a top-down method.

Reinforcement learning is another subfield of machine learning. It focuses on how an agent learns to make sequences of decisions by interacting with an environment to maximize some notion of cumulative reward.

Machine learning is by no means limited to the areas mentioned here—there's a whole lot more that can be said about it; these are just the most common types and applications. We will explore more about machine learning and put what we have learned into practice as we progress through this guide.

Artificial intelligence

The field of AI itself is focused on building systems that mimic human intelligence. This includes tasks such as learning, reasoning, problem solving, perception, and understanding human language. The goal is for AI systems to continuously adapt, improve independently, and support humans in making informed, autonomous decisions—such as assessing risk based on prior data or aiding in complex decision making. It's also important to note that at this stage, we are still working in conjunction with AI—it's meant to be a tool to support us and not act on its own.

Deep learning

Deep learning is a subset of machine learning that uses artificial neural networks with multiple layers to model complex patterns and relationships in data. It has been at the center of many recent advancements in the field of AI, such as natural language processing, the development of autonomous vehicles, and image and speech recognition. Deep learning models utilize a variety of neural network architectures, including feedforward networks for general prediction tasks, convolutional neural networks for

images and spatial data, recurrent neural networks for handling sequences, and transformer networks that employ self-attention mechanisms for advanced language understanding.

Deep learning models have three types of layers: input, hidden, and output. The *input layer* receives the input data. The *hidden layers*, located between the input and output layers, perform most of the computational work; there may be hundreds of hidden layers, depending on the complexity of the deep learning model. The *output layer* produces the prediction or classification, given the patterns that have been extracted and learned.

These models learn by adjusting the weights of connections between nodes through a process called *backpropagation*. During training, the model creates predictions, compares them with the actual outcomes, calculates the error, and adjusts the weights to minimize that error. This cycle repeats continuously, so the network iteratively improves its accuracy. Although the mathematical operations inside every layer are fully transparent, the vast number of parameters and nonlinear interactions makes it difficult to trace exactly why a specific input leads to a particular output. This complexity gives deep networks their "black box" reputation and raises explainability concerns in high-stakes domains such as health care and law, where decision transparency is critical.

Table 1-1 provides a practical comparison of the four aforementioned disciplines and the corresponding Azure Services involved, to help you get a better understanding of which disciplines will be used when.

Table 1-1. Comparisons of the disciplines and relevant Azure Services

Discipline	Definition	Typical Azure services	Typical skillsets engineers use	Example use case
Data science	Used to extract insights from data by using statistics, math, programming, and domain expertise	Azure Synapse Analytics (for data ingestion, transformation, and exploratory analytics),[a] Azure Databricks (for collaborative data engineering, exploratory analysis, and ML pipeline orchestration), and Azure ML (for data preparation, experimentation, and model deployment)	Statistical analysis, programming in Python or R, SQL proficiency, data visualization, and domain expertise	An insurance company analyzing policyholder data in Synapse to discover fraud patterns or optimize premium structures

Discipline	Definition	Typical Azure services	Typical skillsets engineers use	Example use case
Machine learning	Used for training models to make predictions or decisions without explicit programming instructions	Azure ML and Azure AI services (Azure AI Custom Vision and Language)	Machine learning algorithms, feature engineering, Python proficiency, scikit-learn proficiency, model evaluation, and deployment	A retailer building a product recommendation engine in Azure ML and deploying it to a web app
AI	Systems designed to mimic human intelligence (planning, reasoning, perception, etc.)	Azure AI services (Azure AI Vision, Speech, and Language) and the Azure OpenAI Service	AI architecture design, knowledge representation, NLP and computer vision techniques, Python proficiency, and REST API integration	A customer support solution that uses the Azure OpenAI Service to handle routine inquiries via a chatbot
Deep learning	A subset of machine learning that uses multilayered neural networks to learn complex patterns	Azure ML (GPU-based training), the Azure OpenAI Service, and specialized frameworks (PyTorch and TensorFlow)	Neural network architecture, TensorFlow or PyTorch use, GPU optimization, hyperparameter tuning, and Compute Unified Device Architecture (CUDA)	An automotive company training an advanced image recognition model for use in autonomous vehicles on Azure ML's GPU clusters

[a] Since Microsoft Fabric is intended to be Microsoft's future analytics platform, Azure Synapse Analytics will be migrated toward it.

These real-world examples illustrate how each discipline maps to specific Azure services and how organizations commonly employ these technologies to solve business problems.

Now that you understand these fundamentals, we can look at the different considerations for developing AI responsibly in the environment of Microsoft Azure.

The Six Core Principles: Considerations for Developing AI Responsibly

When you're developing AI solutions on Microsoft Azure, it's imperative that you adhere to the established principles and practices that guide ethical AI development. To that end, Microsoft has established six core principles of ethical AI development that you should follow: fairness, reliability and safety, privacy and security, inclusiveness, transparency, and accountability (see Figure 1-2). It's very important to keep these principles in mind while you're learning about AI throughout this book, as they will guide how you implement responsible AI solutions.

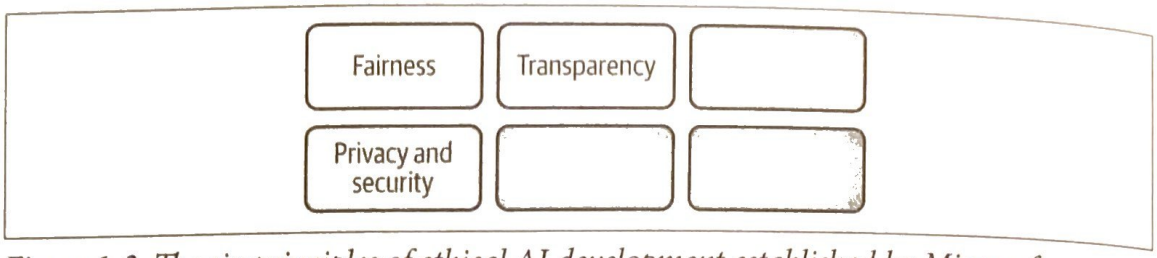

Figure 1-2. The six principles of ethical AI development established by Microsoft

Let's explore what each of these principles stands for:

Fairness
: This principle ensures that AI systems do not discriminate against individuals or groups and that they can allocate resources, information, and opportunities equitably. This is important, given that AI models are only as good as the data used to build and train them. It's easy to unintentionally introduce bias into these models—hence the importance of being aware of this factor.

Reliability and safety
: This principle focuses on a system's performance across different conditions and its resilience to manipulation. In generative AI models, for instance, this is a big issue that's still being addressed, as users often attempt to find ways to ask the AI questions that developers have specifically tried to filter out from being answered.

Privacy and security
: This principle highlights the importance of protecting personal data within AI systems. It ensures that strong data protection measures are in place and that data usage complies with privacy laws and ethical standards. It also extends to ensuring users have transparency into and control over how their data is used in these systems.

Inclusiveness
: This principle dictates that AI must be accessible to and considerate of the diverse needs of all users, including factors such as disabilities and cultural, geographical, and socioeconomic backgrounds. Designing for inclusivity requires creating systems that are user-friendly for a broad audience and actively addressing and mitigating potential barriers that could exclude certain groups from benefiting from AI technologies.

Transparency
: This principle highlights the need to make the mechanisms of AI systems understandable to all relevant stakeholders. It ensures that clear communication is provided on how an AI system makes decisions and uses data, and what the limitations of the system are. Users are often asked to provide personal data, and

transparency is vital to help them make informed decisions about sharing such data.

Accountability
: This principle holds that there should be clear accountability for outcomes provided by AI systems. Organizations and individuals that are involved in the design, development, and deployment of AI technologies must be responsible for ensuring that AI systems adhere to ethical standards and legal requirements. Mechanisms, such as ethical review processes and oversight bodies help maintain accountability and proactively mitigate the risk of AI systems causing harm or making errors.

There are significant risks to not adhering to these principles. A 2019 study[3] in the journal *Science* revealed that a widely used healthcare prediction algorithm, which relied on historical healthcare costs as a proxy for patient risk, systematically underestimated the needs of Black patients. As a result, they received far fewer referrals for additional care despite having similar health profiles to White patients. That's a stark example of how biased proxies can undermine fairness in AI systems.

Beyond memorizing these principles, there's another way to help yourself abide by them: understanding explainable AI (XAI) techniques. We'll cover that next.

Explainable AI Techniques

To make AI systems transparent and interpretable, you can employ several explainability methods during development and deployment. Model-agnostic local explanation techniques such as *Local Interpretable Model-Agnostic Explanations* (LIME) alter input data and fit a simple surrogate model around each prediction to show which features drove a given decision, while *SHapley Additive exPlanations* (SHAP) use Shapley values from game theory to assign fair contribution scores to input features. Gradient-based attribution methods, including *integrated gradients* and *Deep Learning Important FeaTures* (DeepLIFT), trace how incremental changes in each input influence the model's output and generate saliency maps that highlight critical regions or tokens.

Counterfactual explanations create minimal "what if" input variants that would flip a prediction, which helps users explore decision boundaries and understand alternative outcomes. *Concept activation vectors* (CAVs) probe hidden layers by measuring a

3 Ziad Obermeyer et al., "Dissecting Racial Bias in an Algorithm Used to Manage the Health of Populations" (*https://oreil.ly/yZnY6*), *Science* 366, no. 6464 (2019): 447–53.

model's sensitivity to high-level concepts, thus surfacing potential biases or unexpected behaviors. Also, standardized documentation in the form of model cards and datasheets ensures that stakeholders have clear information on intended use cases, training data composition, evaluation metrics, and known limitations.

Finally, Azure ML's Responsible AI dashboard, which is built on the InterpretML package, offers no-code visualizations of global feature importance, local explanations, cohort analyses, and counterfactual scenarios—all integrated into your model lifecycle. These techniques empower developers, data scientists, and business leaders to trust, debug, and refine AI systems by revealing how models arrive at their predictions.

Tech Setup

Before we delve into exploring and learning all you need to prepare for the AI-102 exam, let's configure the foundational components that you'll need to develop AI solutions throughout this book. This will include setting up and configuring your Microsoft Azure Account and the provided AI services, so that you have a playground in which to experiment with them. We'll also cover setting up your local environment for coding and working with these services.

In the following sections, I'll go through each of the components you'll require for this book, and I'll guide you through setting them up.

Setting Up Your Microsoft Azure Account

The first step on your journey to working with AI solutions is to set up your Microsoft Azure account. This will allow you to experiment with all the Azure AI offerings and integrate them into your local environment.

Follow these steps to get set up:

1. Visit the Azure website (*https://azure.microsoft.com*) and click the "Get started with Azure" button (see Figure 1-3). If you already have a Microsoft account (e.g., from using Microsoft 365), you'll be prompted to log in, and some of your details may already be filled in.
2. After signing up, you'll need to verify your account, typically through a phone verification process such as SMS.
3. You will then need to input your payment information. Even for a trial account, Azure requires a valid credit or debit card for identity verification purposes. Note that there's no charge for setting up a trial account, but your bank statement may show a $0 transaction as a record of the verification.
4. After entering your payment information, read the terms and conditions and click Sign Up.

5. Once your account is set up, you can access it by clicking the My Account link in the top-right corner or by going directly to the Azure portal.

Figure 1-3. The landing page for Microsoft Azure

Feel free to poke around the website if this is your first time creating an account. I also recommend learning about billing and setting up governance, security, and compliance according to your needs.

Guidance on these topics will not be covered in this book, but you can read about them on the "Azure governance documentation" page (*https://oreil.ly/aWb30*) in the official Azure documentation. It's also a good idea to become familiar with common quota limits (such as CPU hours and specific region usage) that come with Azure trial or pay-as-you-go subscriptions. If you see errors related to usage quotas or resource creation limits, you can visit the "Azure subscription and service limits, quotas, and constraints" page (*https://oreil.ly/PcCPb*) for information on requesting quota increases. If your phone or credit card verification fails, check that you entered your details correctly and whether your card provider allows $0 verification transactions.

Configuring Your Azure AI Environment

To start using most AI offerings from Microsoft Azure, you'll need to set up Azure AI services. This involves configuring an Azure AI resource. To do that, follow these steps:

1. Locate it on the dashboard or via the search bar, and click Azure AI Services.
2. You will be prompted to create an Azure AI resource if you don't have one yet. In that case, go ahead and create one.
3. When you create an Azure AI resource, your subscription will be set to your default one (see Figure 1-4). You can change this if desired. Then, select the resource group that you would like to use, or create one if you don't have one yet. Name it something that will remind you of what it's used for. (I've chosen to call mine AI102GuideRG.) Pick the region that you would like to deploy in, and choose a name for the resource. (I've called mine AI102Resource to reflect its use.) You can also add a description. When you're ready, click Next.

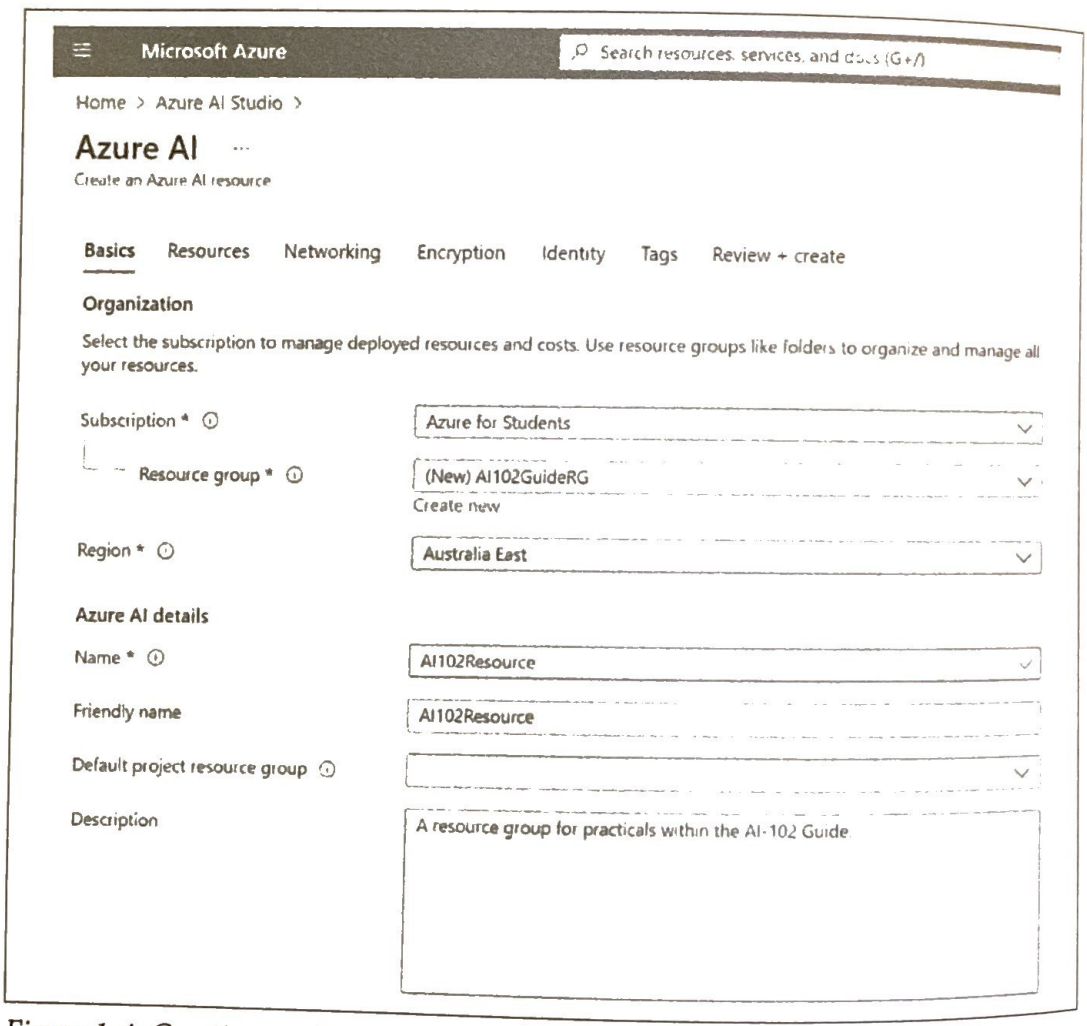

Figure 1-4. Creating an Azure AI resource in Azure AI Foundry

4. Now, you need to configure the associated resources. Most of the fields on the Resources tab will be pre populated for you based on the resource name that you have chosen, but you can choose new names for the Storage Account, Key Vault,

and Application Insights resources by clicking "Create new." When you're ready, click Next.

5. On the Networking tab, you'll see three different options for network isolation (see Figure 1-5). For the purposes of this introductory chapter, we will use Public. (Note that you will be configuring more secure resources in the upcoming chapters based on best practices.) Then, click Next.

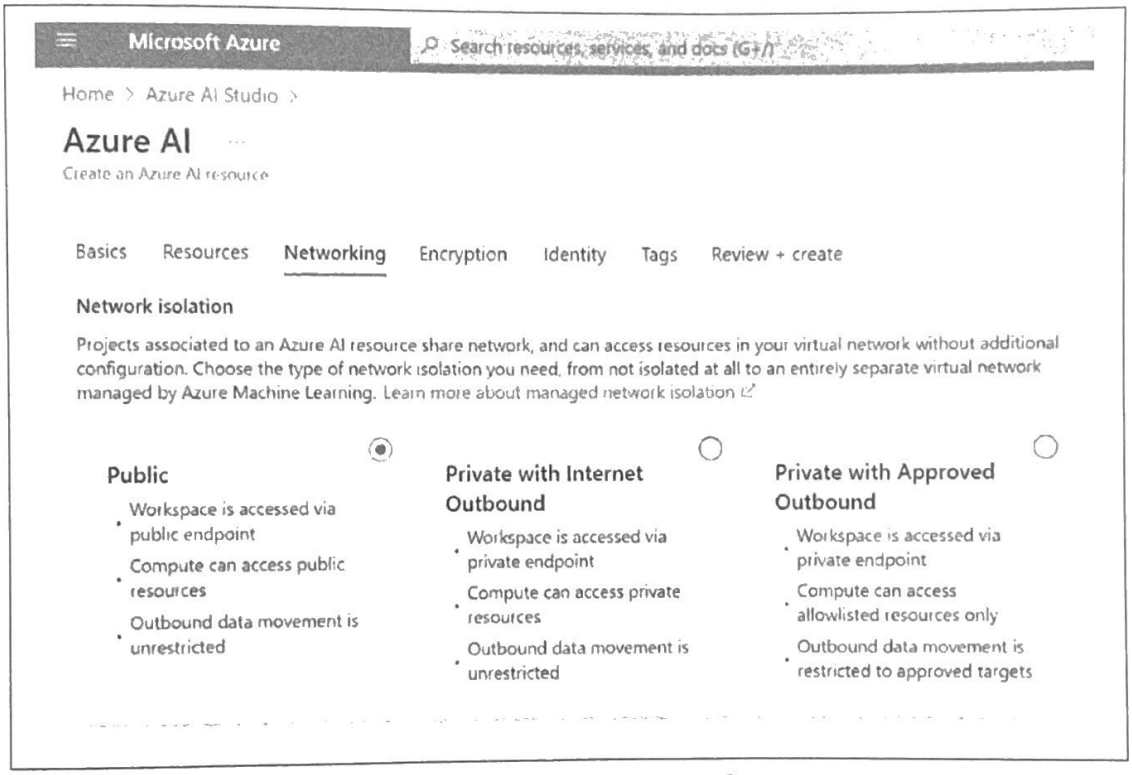

Figure 1-5. Configuring the Networking tab to create the Azure AI resource

6. On the Encryption tab, you can choose whether to use Microsoft-managed keys or customer-managed keys for data encryption. Choose "Microsoft-managed keys" and click Next.
7. On the Identity tab, you will see options for the managed identity type. For the purposes of this configuration, select "System-assigned identity." Then click Next to continue.
8. That will take you to the "Review + create" tab. Review the configurations, then click Create to create the resource.

With that, you will be able to see all the different AI services provided by Microsoft Azure. It may seem like a lot, but I'll walk you through the ones that are necessary for the exam. Feel free to explore the different solutions on the page and find out what each is one used for. Becoming familiar with these services will help you build a

foundation for understanding which workloads to apply in practice and on the exam itself.

Configuring Your Local Development Environment

Next, you'll need to configure your local development environment so you can build AI solutions on Azure throughout this book. This requires installing Python and the necessary libraries for working with Azure.

The steps outlined here are for Windows—they'll be similar if you're using macOS or Linux, although the UI might differ:

1. If you don't have Python yet, you can download it from the Python home page (*https://www.python.org*), where you can also access documentation that will give you instructions on configuring it for your environment.
2. Test that it was installed successfully by running the following command in your command prompt:

   ```
   python --version
   ```

   If Python was installed successfully, you should see the version number as the output of your command (see Figure 1-6). If nothing appears, check that the Python installation path is included in your system's environment variables. Note that the version number you see may differ from what's shown here; it will match the version that you have installed.

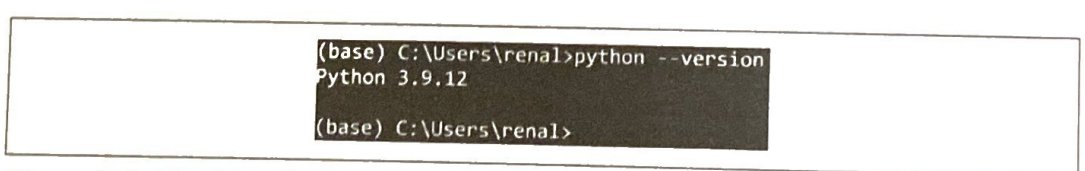

Figure 1-6. Checking the local environment's Python version

3. Install the Azure Command Line Interface (CLI) for your operating system. This is a set of commands used to manage Azure resources. It's essential for creating and managing Azure resources from the command line or through scripts. You can download it and find installation instructions on the "How to install the Azure CLI" page (*https://oreil.ly/sD3pN*) in the official Azure documentation.
4. To log in to your Azure account and verify that your installation has succeeded, run the following command, entering your username and password instead of the placeholder values:

   ```
   Az login -user YOUR_USERNAME_HERE --password YOUR_PASSWORD_HERE
   ```

Note that this is done here for simplicity's sake, but it's not a best practice. The Azure CLI will enforce the use of multifactor authentication (MFA) starting in September 2025; you can read more about this on the "Sign into Azure interactively using the Azure CLI" page (*https://oreil.ly/-N1zy*).

5. Verify that you have a code editor set up for local development. If you don't have one yet, I recommend installing Visual Studio Code (VS Code) as your integrated development environment (IDE) for Python. VS Code supports Python development natively and offers extensions for working with Azure.
6. Install the Python extension for VS Code from the marketplace (see Figure 1-7). This will provide enhanced support for Python, including IntelliSense, linting, debugging, code formatting, code navigation, and refactoring.

Figure 1-7. Installing the Python extension for VS Code

7. Install the Azure Core SDK for Python by using `pip`:

```
pip install azure-core
```

The Azure Core SDK for Python provides the necessary libraries for a myriad of Azure services that are core to utilizing the platform, which you can use to work with Azure resources directly from your Python code. We'll be downloading more libraries along the way.

8. Next, install the Azure Identity library, which will allow you to authenticate with Azure services. It will provide you with support for different authentication methods, such as Microsoft Entra ID and managed identities. This will also be very helpful to you, because you'll be using authentication throughout your experimentation with Azure. You can install the library using `pip` as follows:

```
pip install azure-identity
```

9. It's good practice to use virtual environments for Python projects to manage your dependencies. This helps keep certain dependencies of one project from interfering with the dependencies of another. You can create a virtual environment for this chapter with the following command:

```
python -m venv /your/path/to/the/virtual/environment
```

Here, */your/path/to/the/virtual/environment* is the location where you want to create your virtual environment.

For compatibility, it's recommended that you use Python 3.8 or later when working with the Azure SDKs. Some older versions of Python or older Azure SDK releases may not support newer service features. If you encounter module import errors or version conflicts, try upgrading via `pip install --upgrade` *package_name*, or consult the official Azure SDK for Python documentation for specific version requirements.

If your local environment commands fail (e.g., if the `az` command is not found), verify that the Azure CLI has been correctly added to your system's PATH and that you have reopened your terminal or command prompt after installation. If you're using Windows, you may need to restart the system or log out and back in again for changes to take effect. On resource-constrained machines, you may also run into out-of-memory errors when generating large SHAP or LIME explanations—in that case, you should reduce the sample size for explanations or increase your compute target's available RAM.

With that, you should have a local development environment that's configured and ready to go! We can now move on to exploring the capabilities of Azure AI.

Gaining an Understanding of Azure AI's Capabilities

Building AI solutions on Azure is less about mastering every tool and more about knowing which tool stops the bleeding fastest. Picture this: you're tasked with creating a chatbot for a hospital. Do you grab Azure's prebuilt Health Bot, which will save you six months of Health Insurance Portability and Accountability Act (HIPAA) compliance headaches, or do you stubbornly train a custom model and risk leaking patient data?

The AI-102 isn't a trivia quiz—it's a crash course in triage. We'll start by mapping Azure's AI services like a paramedic's kit, learning when to use Azure AI Computer Vision for X-ray analysis and when to use Azure AI Custom Vision to spot defective IV bags on a factory line. You'll learn why deploying a speech-to-text model locally with Docker can slash latency for 911 call analysis, and you'll learn how to avoid making the rookie mistake of using GPT-4 for simple FAQ bots. By the end, you'll stop asking the question, "What does this service do?" and start answering the question, "How do I ship this without getting fired?"

The Capabilities of Microsoft Azure's AI Services

Learning about all of Microsoft Azure's AI service offerings may seem like quite a hurdle, given how many there are, but they all fit into a few categories based on their capabilities. Let's explore these capabilities.

Machine learning

As mentioned previously, machine learning is a branch of AI that provides systems with the ability to autonomously learn and improve from experience without being explicitly programmed by humans. It involves developing algorithms that can analyze and learn from data, creating models, and running them to make predictions or decisions based on that data. Azure offers the Azure ML service, which helps streamline the process of developing, training, and deploying ML models at scale.

The machine learning workflow is made much easier by the suite that's provided with Azure ML. Users can use automated machine learning to build models more efficiently, and it simplifies the process by automatically selecting algorithms and tuning parameters, thus allowing for continuous improvement. Azure ML's user interface (UI) makes this simple even for nontechnical people.

The workflow is complemented by the Azure Machine Learning Designer, which is geared toward those who do not have extensive development experience. It provides a no-code environment for the development of ML solutions, and many of the features you'll find there are drag-and-drop. You can therefore immediately get to work and focus on getting the model running based on your own requirements, instead of spending time setting up your coding environment.

In terms of practical limits and performance, the maximum number of parallel training jobs, central processing unit/graphics processing unit (CPU/GPU) quotas, and supported virtual machine (VM) sizes can vary by subscription level and region. For instance, some regions may not support specialized GPU clusters, so check Azure ML region availability (*https://oreil.ly/fd_z0*) for up-to-date information on quotas.

Computer vision

Computer vision is a branch of AI that enables applications to interpret and understand visual content in images and videos in a way that mimics human vision. The Azure AI Vision service provides most of the capabilities of this branch in Azure, allowing for the implementation of diverse workloads, such as analyzing images, reading text, and detecting faces.

Like other cognitive services, Azure AI Vision has specific regional availability and usage tiers. For instance, the free tier may limit the number of transactions (calls per second) and the size of images or videos processed. A higher-paid tier typically provides better performance throughput and larger file size allowances.

This branch presents several different functionalities, such as:

Image classification
: This involves classifying different types of images into different categories. For example, it could include classifying objects differently based on whether they are green or orange.

Object detection
: This capability helps with locating and identifying different objects within images by extracting attributes such as their types and positions. This helps in applications where you want to find a particular object in each image.

Semantic segmentation
: This involves categorizing the pixels within an image into a specific class or object. Unlike object detection, which identifies and locates objects in an image, it classifies on the pixel level, thus ensuring that each pixel is labeled as belonging to a specific class. Semantic segmentation is applied in an array of industries. For example, it's used to help autonomous vehicles understand their driving environment by segmenting roads, pedestrians, vehicles, and obstacles with pixel-level precision, thereby enhancing safe navigation.

Image analysis
: This capability helps with identifying objects and applying tags and descriptions within images to provide insights about their content.

Face detection, analysis, and recognition
: This involves recognizing human faces and making predictions of attributes such as age, gender, and emotions based on the data that the model is trained on. It's also useful for authentication purposes—an example is the Face ID capability that you can unlock your mobile phone with by simply pointing it at your face.

Optical character recognition

Optical character recognition (OCR) involves the extraction of text from images and documents in a multitude of different languages. The images recognized can contain handwriting or printed text. This is very useful in quickly scanning documents to store or manipulate digitally without the need to retype them. Many third-party services—such as Adobe Scan, which scans images and saves them digitally so you can edit them later—utilize this capability.

Document intelligence

Document intelligence is a branch of AI that is used in understanding, analyzing, and extracting meaningful information from documents. It is usable with various types of documents—including PDFs, images, emails, and scanned texts—to convert unstructured data into structured, actionable insights. Microsoft Azure has Azure AI Document Intelligence to help streamline this process: it can be integrated into applications and workflows to extract information from different sources, such as invoices and forms.

In practical terms, the Azure AI Document Intelligence service has limits on document size, page count, and calls per second. For instance, certain tiers may process only up to 200 pages per document. Be sure to verify region support and capacity requirements if your documents are large or numerous.

Knowledge mining

Knowledge mining is a branch of AI that helps with the extraction of information from a massive amount of data (either structured or unstructured), to construct a knowledge store that can be searched. Azure AI Search is a knowledge mining solution that's provided by Microsoft Azure. You can use it to build indexes that make searching documents easier or to enhance searching of documents that are publicly available on the internet.

Although knowledge mining may sound like document intelligence, the two branches of AI differ in a few areas. In terms of scope, document intelligence primarily focuses on the extraction and analysis of data from documents, while knowledge mining encompasses a broader scope of diverse data types and sources beyond just documents. The goals of the two branches also differ: document intelligence primarily aims to convert unstructured document data into a structured format for easier analysis and automation, while knowledge mining focuses on uncovering hidden insights and relationships across a wide array of data to build a comprehensive knowledge base.

Natural language processing

Natural language processing (NLP) is a branch of AI that focuses on enabling computers and humans to interact through natural language. It aims to help computers understand, interpret, and produce human language in an interpretable way. This includes being able to understand printed text in files such as documents and emails, interpret speech and reply appropriately, and translate text in a given language into other languages.

Microsoft Azure offers a fair number of services that help with this. Azure AI Language, for instance, helps in building NLP solutions that process and analyze text, and Azure AI Speech enables solutions to understand human speech and transcribe it or generate a reply based on what was said. NLP also makes up a big chunk of generative AI, particularly in the area of understanding what is put through and ensuring that the response from the foundational model is appropriate.

Generative AI

As I touched on briefly earlier, generative AI encompasses a variety of AI technologies that specialize in creating original content. Its most common application is integration with chat applications, allowing them to process inputs in natural language and output responses in different formats, such as text, images, code, and audio.

With Microsoft Azure, we can develop generative AI applications using Azure OpenAI. This service makes use of the generative AI capabilities of OpenAI's models and APIs and Azure's infrastructure, which helps it integrate with other services within Azure and allows for scaling and securing it appropriately. Through the Azure AI Foundry interface, you can work with the different foundational models that are provided to adapt generative AI to your use case.

That completes our overview of the different capabilities that Microsoft Azure AI services have to offer. The rest of this guide includes separate chapters on all of these capabilities, where we'll delve deeper into them while giving you a chance to get hands-on experience and apply them to your own use cases. Figure 1-8 is a decision tree that can help you determine which service matches the specific requirements of any given scenario.

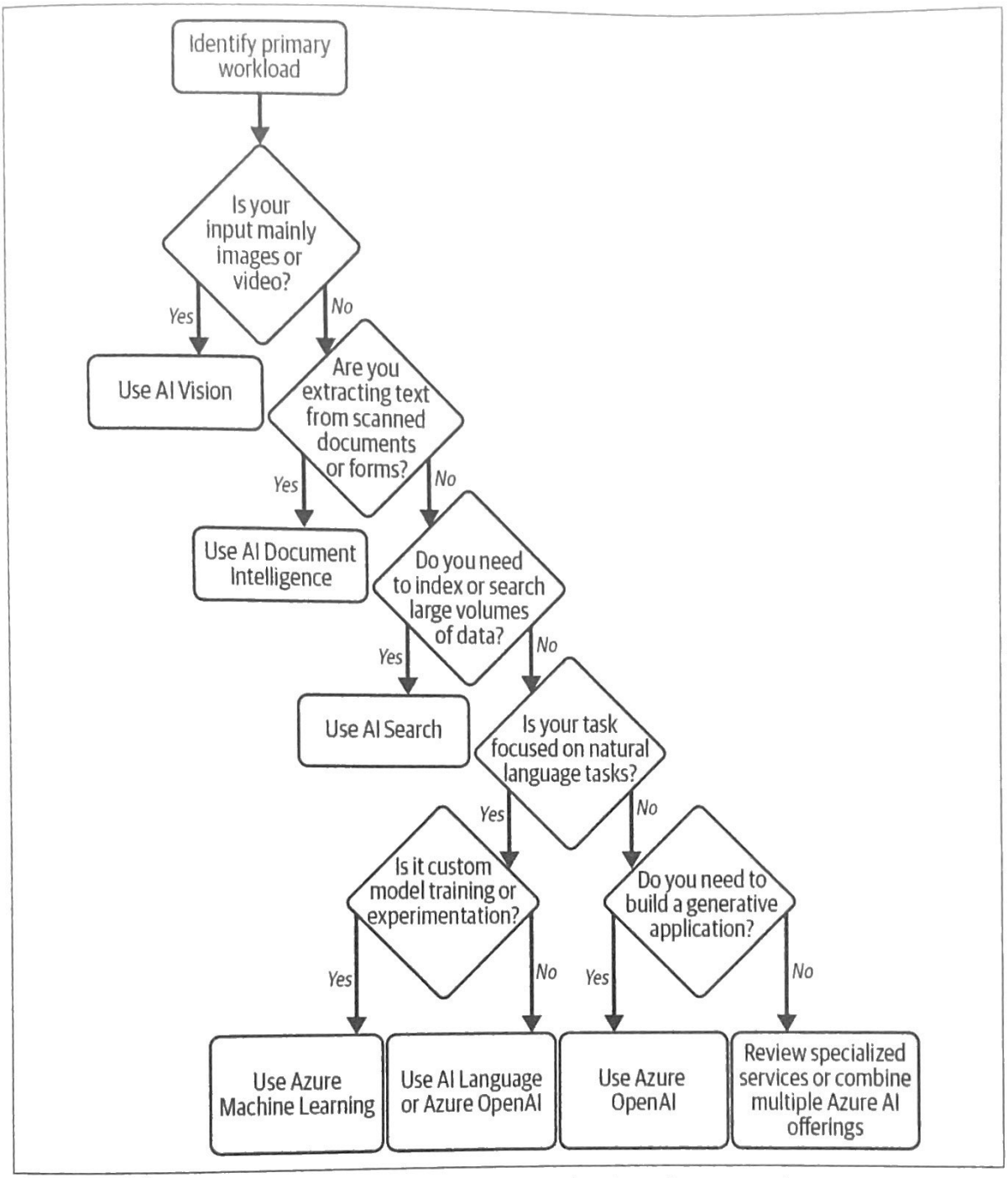

Figure 1-8. A decision tree to identify the service that best fits a specific scenario

Due to the high computational requirements of large language models (LLMs), Azure OpenAI is available only in certain regions that have specialized infrastructure. Usage is billed based on tokens processed or images generated, and there are often rate limits for free or preview accounts. Check the "Azure OpenAI Service pricing" page (*https://oreil.ly/UbGIq*) for the current cost per 1,000 tokens.

Table 1-2 provides a comparison of Azure's various machine learning, computer vision, and document intelligence services to help you understand more about their practical limits, performance expectations, and regional availability.

Table 1-2. A comparison of Azure AI services

Service	Practical limits	Performance expectations	Regional availability	Key features
Azure ML	Compute resources affect performance, and large datasets require optimized storage.	Performance scales with compute resources, and AutoML simplifies model training.	It's available globally, but certain features are limited by region.	Automated ML, no-code Azure ML Designer, machine learning operations (MLOps) pipelines, model registry, and experiment tracking
Azure AI Vision	Request size limits apply, and high-resolution images may require resizing.	Throughput varies with request volume, and batching improves performance.	Most features are available globally, but some advanced features are region specific.	Image classification, object detection, optical character recognition, face detection, and spatial analysis
Azure AI Document Intelligence	There are maximum file size and page count constraints, and large documents require splitting.	Higher pricing tiers offer increased processing speed.	It's available in multiple regions; refer to the Azure documentation for specifics.	Form recognizer, invoice and receipt extraction, table and key-value extraction, and custom model training
Azure AI Search	Index and data source limits depend on pricing tiers.	Optimized indexing improves search speed, and higher pricing tiers support larger data volumes.	It's available in multiple regions, but some advanced features are region-specific.	Full-text and semantic search, vector search, faceted navigation, autocomplete, and scoring profiles
Azure AI Language	Text size and document count per request have limits.	Speech recognition accuracy depends on audio quality, and batching improves text processing.	It's widely available, but certain features are limited by language and region.	Sentiment analysis, key phrase extraction, entity recognition, translation, and summarization
Azure OpenAI Service	There are quotas on requests and token processing per minute.	Provisioned throughput units (PTUs) enhance generative model response times.	There's limited availability, and regional access restrictions apply for some models.	Text and code generation, embeddings, fine-tuning, multimodal input, and chat/completion endpoints

Consuming AI Services

There are three main ways to consume AI services: through the Microsoft Azure UI, by using SDKs, and by using REST APIs. In this section, we'll discuss all three of these ways.

Microsoft Azure user interface

The Azure portal provides a user-friendly GUI for accessing and managing Azure AI services. This approach is great for users who prefer a visual, point-and-click experience over writing code. Such users can directly configure AI services in the relevant AI workspaces through the portal, and they can also test the services and deploy them directly from the portal. Alongside this, they can monitor the performance and usage of AI services while managing access and security settings accordingly. However, since you'll need to integrate AI services into your own code and web applications, using the GUI will often be a limiting option for you. Therefore, the main method of consumption we'll discuss in this guide will be through SDKs.

SDKs

SDKs provide a more streamlined way to interact with Azure AI services through predefined libraries. They're provided for numerous popular programming languages, such as Python, .NET, Java, and Node.js. Developers can integrate SDKs into their applications as libraries, allowing them to use predefined classes and methods that facilitate interaction with AI services. This abstracts away much of the complexity involved in making HTTP requests and parsing responses, compared to directly calling REST APIs.

REST APIs

Using REST APIs is a flexible, language-agnostic way to integrate with Azure AI services. Users send HTTP requests to the AI service's endpoint, which includes the necessary headers and optional payload. The requests must be authenticated using keys or tokens that are obtained during the configuration of the service. The service will then process each request and return the response in a standard format (normally JSON), which includes the results based on the operation requested. The main benefit of this method is that it can be integrated into any application or script that can make HTTP requests, meaning it offers you a high degree of flexibility in terms of which programming language and environment you use. However, this method is also more complex because it requires you to understand how to make such requests and doesn't use predefined methods like SDKs.

As mentioned previously, we'll mostly use SDKs in this guide, given their ability to flexibly integrate with Python. However, we'll still discuss a few examples that use REST APIs to help you understand how they are used.

Authentication and Security

In Microsoft Azure, authentication is a critical aspect of securing your AI applications so only authorized users can access them. There are three key methods of authentication: Microsoft Entra ID (Azure ID), subscription keys, and managed identities. In

this section, we'll discuss each of these methods and what they mean for your own AI solutions.

Microsoft Entra ID (formerly known as Azure AD)

This is an identity and access management service based on Azure that provides authentication and authorization to Azure services. It allows for single sign-on (SSO), which lets users access multiple services with one set of credentials; multifactor authentication, which requires two or more verification methods to authenticate (such as password and SMS); and role-based access control, which allows for access management based on user roles.

Subscription keys

Using subscription keys is the most common method of authentication with Azure AI services. When an AI service is created, Azure generates two keys for the resource, and the keys are then used alongside the endpoint to authenticate API requests by including them in the request headers. This is the main method we'll use to authenticate AI services throughout this guide. It's a straightforward authentication process, and it's made even simpler by SDKs, which we'll be using for development.

Managed identities

These provide Azure services with an automatically managed identity within Microsoft Entra ID, allowing users to have secure, simplified authentication in Azure without having to manage their credentials explicitly. Managed identities can be either system-assigned or user-assigned. System-assigned identities are directly tied to an Azure service instance, and when the service is deleted, the credentials and the identity are automatically cleaned up. User-assigned identities, on the other hand, are standalone Azure resources. They can be assigned to one or more Azure service instances, thus providing centralized management of identities.

More security measures

Real-world best practices often involve storing subscription keys or other secrets in Azure Key Vault, rather than embedding them directly in code or configuration files. Azure Key Vault provides a secure way to manage and automatically rotate keys, thus ensuring minimal exposure if credentials are compromised. For example, an AI application might retrieve its subscription keys at runtime from Azure Key Vault by using a managed identity, thus avoiding the use of any plain-text credentials altogether.

You may also configure conditional access policies within Microsoft Entra ID to restrict sign-in attempts from unknown or risky locations. This is especially important in enterprise scenarios where production AI services must be accessed only from certain networks or via a virtual private network (VPN).

In the next chapter, we'll have a more in-depth discussion of securing your environment and what that means for your AI workloads.

Billing and Cost Management

To successfully develop AI solutions on Microsoft Azure, you'll need to gain a well-rounded understanding of how billing works. That will help you manage costs effectively, as AI system development can become expensive if not carefully planned. As an AI engineer, you can also recommend solutions that balance performance and cost efficiency.

Azure's AI services follow a consumption-based pricing model, which means that you pay for what you use without having to account for up-front costs. This lets you flexibly scale services based on your needs—the amount you will be billed for will usually depend on the number of transactions you process, the execution time, and the volume of data you process.

Microsoft Azure provides several tools to help you track and manage the costs of your AI workloads. One of these is Microsoft Cost Management, which enables you to monitor and analyze how you are using your AI services to better understand where costs are incurred. You can set up budgets and alerts to stay on top of your spending and avoid unexpected charges, and the alerts will warn you if you are going over budget.

Part of good cost management is cleaning up your resources when you're done using them. To help you get in the habit, be sure to do this each time you finish a practical exercise in this guide. Doing that here and in your career will help you avoid forgetting about resources you no longer need so they don't end up being continuous sources of costs.

This concludes our overview of Microsoft Azure's capabilities for building AI systems. This is not a comprehensive look into the full potential of Azure, but it does highlight the important resources that will be relevant to you when taking the AI-102 exam.

Now, before we start building with Microsoft Azure's AI services, we need to finish exploring your responsibilities as an AI engineer.

Your Responsibilities as an AI Engineer

As an AI engineer, your role goes beyond just developing and deploying AI models. You're also entrusted with a broader spectrum of responsibilities that ensure the ethical, sustainable, and user-centric deployment of AI technologies. In this section, we'll discuss the key considerations.

Meeting Challenges and Managing Risks

There are several inherent challenges and risks involved in AI development. You'll need to deal with technical and scalability concerns, security and compliance issues, ensuring resource availability, and ethical and legal issues.

Technical and scalability challenges

These are the main hurdles you will face in AI development. This should come as no surprise—it's a major challenge to manage large volumes of data while ensuring it is of high quality and preparing it for ingestion by AI models. As mentioned earlier, an AI model is only as good as the data that's used to build and train it. If it's not fed the right data, it can become biased and perform poorly. This means you have to make an effort to obtain the right data and process it properly.

Aside from data issues, choosing the right AI models and algorithms that suit the specific needs of a project can be daunting. You must properly balance the complexity, accuracy, and performance of a model. Designing systems that scale efficiently is another key challenge, as model training and usage will not always remain constant. Azure services must be configured appropriately to handle different system loads, especially as applications scale. To meet these challenges, you'll need to have experience in designing and implementing solutions.

Security and compliance

The rise of generative AI has brought security and compliance challenges to the forefront. It's vital to protect sensitive data that's used by AI systems, and organizations must comply with relevant regulations such as the General Data Protection Regulation (GDPR)—a comprehensive data protection law in the European Union that sets stringent guidelines for the collection, storage, and processing of personal information—and the California Consumer Privacy Act (CCPA). Such laws aim to protect individuals' privacy and data rights. While they can complicate the design and implementation of our systems, adhering to them is essential for maintaining public trust and fulfilling our commitments to transparency and the proper safeguarding of users' information.

Resource availability

Resource availability is another big challenge. Access to high-performance computing resources is often necessary for training complex AI models, and ensuring that these resources are always available and efficiently utilized can be both difficult and costly. This is why only large corporations such as Amazon, Google, and Microsoft can run some of the best foundational generative AI models, while other companies use those models without provisioning their own. Creating custom versions of such

models is not only very costly but also requires a great amount of expertise, so it's far more efficient for smaller companies to leverage models provided by others.

Ethical and legal issues

Ethical considerations are at the heart of responsible AI engineering. Engineers must adhere to the principles discussed in "The Six Core Principles: Considerations for Developing AI Responsibly" on page 9 and they must also ensure that their systems are designed and deployed in a way that respects human rights and values and promotes the well-being of all stakeholders.

Most ethical risks in AI stem from data privacy breaches and the consequences that AI systems may have on individuals and society. We've seen this in several cases, such as the lawsuit that famous authors filed saying generative AI systems were trained on their work,[4] which could be an infringement of copyright. Regulations governing advances in this technology are still very fresh, and many AI-related issues fall into legal gray areas.

All of the aforementioned issues will continue to evolve in the future. Properly performing risk management when creating AI systems will help you identify key risks and plan how to mitigate them.

Continuous Learning and Collaboration

AI is an evolving field, and to be a successful engineer, you'll need to have a commitment to continuous learning so you can stay current with the latest developments, technological best practices, and regulatory frameworks. You'll also need to develop great collaboration skills so you can work with peers in interdisciplinary teams and engage with the broader AI community to foster innovation, share knowledge, and tackle complex challenges. A collaborative approach is required not only with fellow AI professionals but also with policymakers and ethicists, to ensure that the AI technologies you develop will serve the best interests of the public.

User-Centered Design

Finally, your responsibilities as an AI engineer include placing users at the center of the AI systems you design. Prioritizing their needs, preferences, and values helps ensure your AI solutions are usable, accessible, and relevant. Throughout the process of designing these systems, you'll need to continuously engage with users, conducting needs assessments, performing testing, and gathering feedback. That will help you

4 Max Zahn, "Authors' Lawsuit Against OpenAI Could 'Fundamentally Reshape' Artificial Intelligence, According to Experts" (*https://oreil.ly/Cye1A*), *ABC News*, September 25, 2023.

make sure that the AI systems you build are both valuable to end users and technically sound.

Now that we've covered your responsibilities as an AI engineer, let's start getting into the nitty-gritty of building with Microsoft Azure's AI services!

Practical: Running a Text Analytics Service

The Azure AI Language service provides many ways for AI to recognize attributes of conversational language and provide insights into or responses to it. In this practical exercise, you'll employ Language for one of its key use cases: sentiment analysis. You'll also gain an understanding of how to process and interpret natural language data, and you'll familiarize yourself with the process of integrating Azure AI services into your applications.

Here are the steps to follow:

1. Set up a Language endpoint. To do this, navigate to the Azure AI Language view via the Microsoft Azure portal (*https://oreil.ly/P3Mbi*) and select "Create" on the "Language Service" screen.
2. That will take you to a screen where you can select the custom features you want (see Figure 1-9). Select the feature titled, "Custom text classification, Custom named entity recognition, Custom summarization, Custom sentiment analysis & Custom Text Analytics for health."

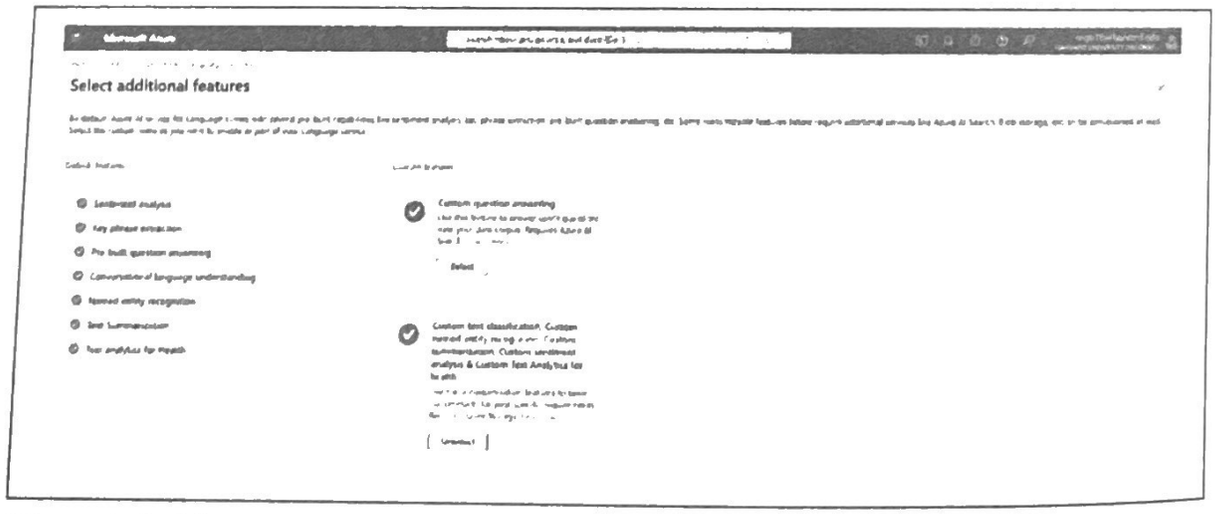

Figure 1-9. Selecting custom features for the Azure AI Language service

3. On the Basics tab on the next screen, select your subscription, resource group, and region as before. Enter a name (e.g., AI102SentimentAnalysis) that you can recognize easily as the instance name, and select Free F0 as the pricing tier. Then click Next.

4. On the Network tab, for the type, choose "All networks, including the internet, can access this resource." Security is an important aspect of AI, but for the purposes of this introductory practical exercise, we'll keep things simple. Click Next to continue.
5. On the Identity tab, select On as the status so you can grant the resource access to other resources. Leave the user-assigned managed identity blank and click Next.
6. On the "Review + create" tab, review your selections and click Create. Your Language instance will now deploy.
7. After the instance has deployed successfully, click "Go to Resource" and choose "Keys and Endpoint" in the Resource Management section of the left pane (see Figure 1-10).

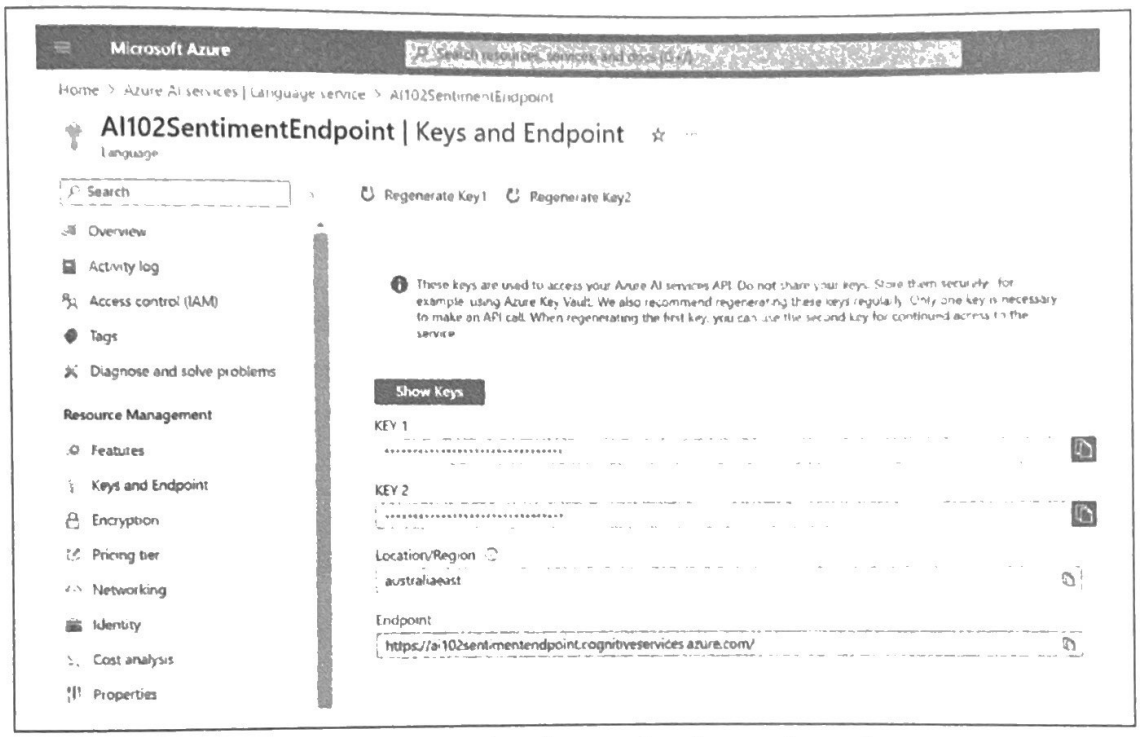

Figure 1-10. Obtaining the keys and endpoints for the configured Azure AI Language service

8. Take note of one of the keys listed there and the endpoint; you will need them later. There are two keys to allow for seamless key rotation so you can regenerate one key while the other keeps your apps running without downtime.
9. Run the following command in your VS Code terminal to install the Azure AI Text Analytics SDK:

```
pip install azure-ai-textanalytics==5.3.0
```

Verify that this is the latest version of the SDK in case there are any updates, and consult the official documentation for any breaking changes when upgrading.

10. Next, you'll start coding a solution in your local environment to call the Text Analytics service. In your VS Code editor (or equivalent), choose File → New File and enter an appropriate name for the file, such as *languagesentimentanalysis.py*.
11. Enter these lines to import the necessary libraries:

```
from azure.ai.textanalytics import TextAnalyticsClient
from azure.core.credentials import AzureKeyCredential
```

12. To authenticate the Text Analytics client, enter the following code:

```
def authenticate_client():
    key = "your_text_analytics_key"
    endpoint = "your_text_analytics_endpoint"
        return TextAnalyticsClient(
            endpoint=endpoint,
            credential=AzureKeyCredential(key),
        )
```

Replace with *your_text_analytics_key* and *your_text_analytics_endpoint* with the key and endpoint values you noted in step 8.

This use of hardcoded subscription keys is acceptable for learning purposes or as a proof of concept, but in a production environment, you'll want to use Azure Key Vault or one of the other methods covered in "Authentication and Security" on page 25.

13. Create a function to analyze sentiment with this code:

```
def sentiment_analysis(client):
    # Prompt the user to enter a sentence
    try:
        user_input = input("Enter a sentence to analyze sentiment: ")

        # Analyze the sentiment of the user's input
        response = client.analyze_sentiment(documents=[user_input])[0]
        print("Sentiment analysis result:")
        print("Overall sentiment:", response.sentiment)
        print(
            "Scores: positive={0:.2f}; "
            "neutral={1:.2f}; "
            "negative={2:.2f}".format(
                response.confidence_scores.positive,
                response.confidence_scores.neutral,
```

```
                response.confidence_scores.negative,
            ))
        except Exception as e:
            print("An error occurred while analyzing sentiment:", e)
```

14. Define the main function and call it like this:

```
def main():
    client = authenticate_client()
    sentiment_analysis(client)

if __name__ == "__main__":
    main()
```

    Inside `main`, you call the `authenticate_client` function and save it to a client to perform the sentiment analysis you require.

15. Run the code in the VS Code terminal with the following command:

```
python languagesentimentanalysis.py
```

16. At the prompt, enter a sentence upon which to perform a test run of sentiment analysis. For instance, I will enter "Hello I am quite happy."

    You should then see the results in the form of overall sentiment and scores in three categories: positive, neutral, and negative (see Figure 1-11). The results for my test sentence show that it has been classified as 98% positive and 2% neutral —so the overall sentiment is classified as positive.

```
Enter a sentence to analyze sentiment: Hello I am quite happy
Sentiment analysis result:
Overall sentiment: positive
Scores: positive=0.98; neutral=0.02; negative=0.00
```

Figure 1-11. Test run of sentiment analysis

17. Experiment with different inputs, deliberately testing negative and neutral sentences to see if the sentiment analysis continues to be accurate.

18. After completing this exercise, clean up the resources you have provisioned to save costs. Refer to Microsoft's "Delete resource groups" (*https://oreil.ly/lqG3F*) instructions for guidance.

In a production environment, you may want to enable monitoring and logging by using Azure Monitor or Application Insights to track usage metrics, errors, and latency. This will also help you identify performance bottlenecks and unusual spikes in sentiment calls. If you expect high volumes of sentiment analysis requests, you can scale out the underlying Language resource tier or use multiple instances behind a load balancer.

To manage costs effectively, consider using free tiers for development and testing, and then carefully monitor your usage with Azure Cost Management. You can also implement a retry policy (e.g., exponential backoff) when the service is rate-limiting requests or experiencing transient failures; this will help ensure your application remains robust under heavy loads.

And with that, you have used your first Azure AI service and integrated it into your environment! You've created an AI instance, used the SDK, and thought about which workload to apply to a particular AI problem. This may have seemed like a simple example, but it has many use cases in industry and forms the cornerstone of numerous applications. Gaining a basic understanding of how Azure AI works and how to integrate it into your own environment will serve you well—and you'll work with it plenty more as we progress through the chapters of this guide, applying these learnings to more complex scenarios.

Chapter Review

In this chapter, you learned about the landscape of AI in the wider world and specifically within Microsoft Azure. You set up your environment on Microsoft Azure and your local environment so that it's ready for you to develop AI solutions. You also gained a high-level understanding of the capabilities of Microsoft Azure for AI development, and you learned how to consume Azure's services for your use cases. On top of that, you gained practical experience using your first Microsoft Azure AI service and integrating it with a Python app running in your local environment. That experience will be helpful as you continue working with other Azure AI services and integrating them into your applications.

Before you take the AI-102 exam, you'll need to familiarize yourself with Python or a similar programming language used in AI development, and make sure you understand the following key points covered in this chapter:

- What REST APIs and SDKs are, and how they help users consume Microsoft Azure AI services
- What AI is and how it differs from related fields
- The general capabilities of Microsoft Azure and when to use each workload for different purposes
- The challenges and risks involved in developing AI solutions
- The principles of responsible AI and how they apply to the work that you do in building AI systems

In the next chapter, we'll look at planning and managing AI solutions in Azure. We'll further explore the AI project lifecycle and how to configure for security and monitor the solutions you create.

Chapter Quiz

1. What is the primary goal of AI?
   A. To enhance human physical abilities
   B. To create autonomous robots
   C. To replicate human intelligence in machines
   D. To replace human tasks with tasks performed by machines
2. Which of the following is a type of machine learning in which the model learns to make sequences of decisions by interacting with an environment to maximize some notion of cumulative reward?
   A. Supervised learning
   B. Unsupervised learning
   C. Reinforcement learning
   D. Transfer learning
3. Which of the following disciplines uses artificial neural networks with multiple layers to model complex patterns and relationships in data?
   A. Deep learning
   B. Data science
   C. Reinforcement learning
   D. Regression analysis
4. What Azure service is primarily used for building applications that can understand human languages in the form of provided text?
   A. Azure AI Bot Service
   B. Azure AI Search
   C. Azure AI Language
   D. Azure Machine Learning
5. Which Azure service is best suited to extracting information and insights from large amounts of unstructured data, such as forms?
   A. Azure Blob Storage
   B. Azure AI Document Intelligence