DigitalOcean

The Ultimate Guide to Building a Solid Tech Stack for Startups and SMBs

DigitalOcean

The importance of a solid tech stack cannot be overstated. As builders decide which technologies to implement for their projects, they're crafting the foundation for their business.

These choices can impact the reliability and scalability of your application, future costs and budgetary decisions, and ultimately affect the speed of growth for your business.

In this guide, we'll take you through the basics of building a tech stack for your backend development and infrastructure needs, including overviews of individual components, things to consider when evaluating providers, and some examples of how other startups have done it.

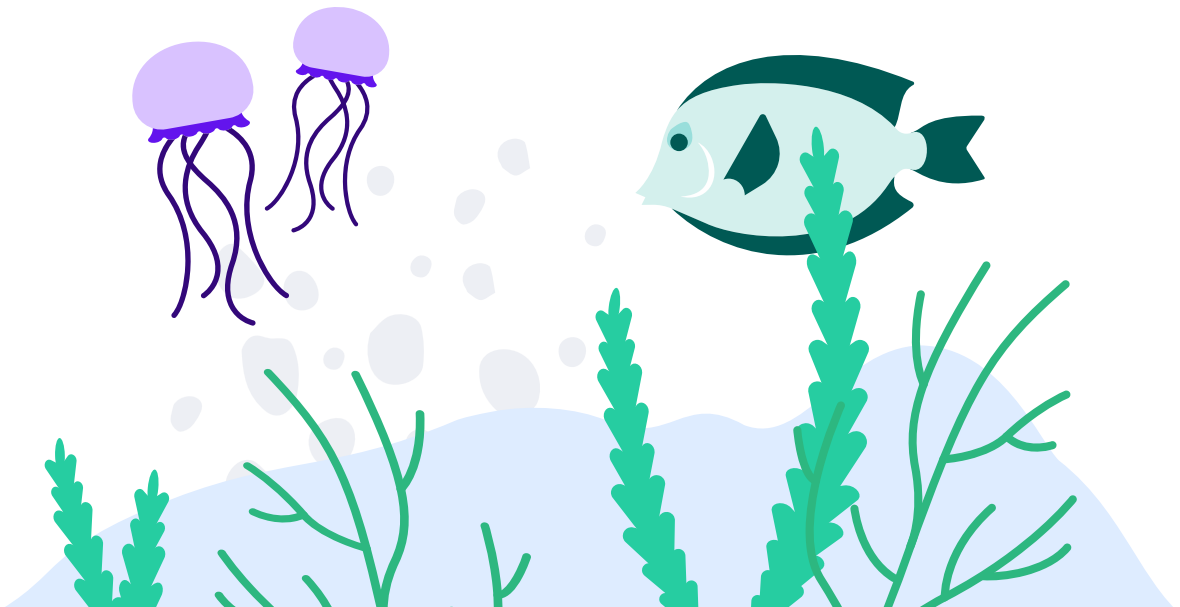
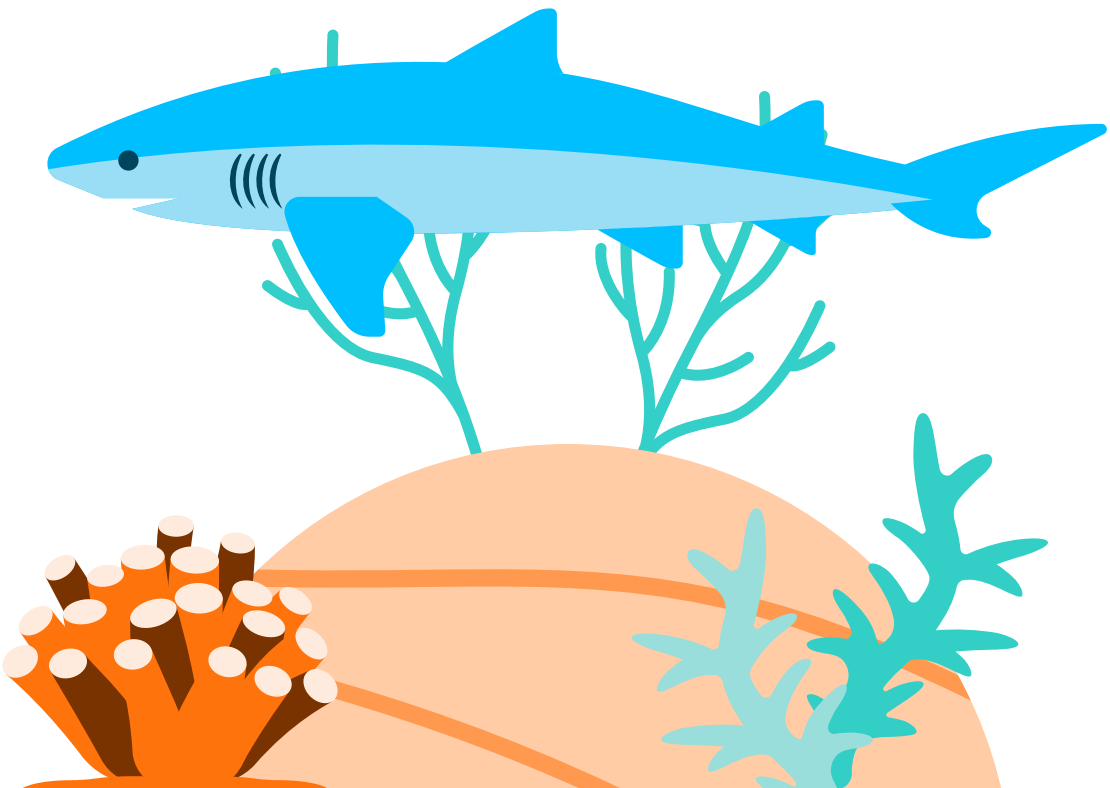


Table of contents



Make your tech stack work for you

Start by understanding your unique needs and goals.

A tech stack is a group of technologies that work together and allow you to build and maintain your application.

Many components need to work together to get the results you want, both on the frontend and backend of the application.

Frontend development takes care of the things that the end-user sees and uses—tools like HTML, CSS, and JavaScript. Backend development is the development of services that support the frontend of your application. This can include anything from calculations, Data Science/ML Models, to database operations. Infrastructure, programming languages, and databases are all components of backend development, and there are many different platforms and frameworks to consider.

Start with an evaluation of your business needs and goals and the strengths and weaknesses of your team. Initial choices while building your tech stack will influence the developers on your team, costs, and time to market. By carefully considering what your team needs from your tech stack, you can avoid troublesome changes in the future.

What is the goal of your application?

Start with the user experience.

What do you want the end-user to see and do?

Are you in need of a static website or an application with decision-making capabilities and many personalized features?

Will there be user accounts or preference centers?

Is this a web, desktop, or mobile application?

The type of application you're building and the functionality you need will affect choices like programming language, data management systems, and more. Making significant changes to your tech stack down the road can add considerable costs or slow down development. Do your best to account for future needs while maintaining simplicity for now.

What are your security needs?

What types of data will you be collecting and maintaining?

Are there industry regulations you need to consider or security standards you need to follow?

What's your budget?

What is your budget now, and how will that change through proof of concept and growth?

Technologies that are less common or difficult to learn will be more expensive to maintain because of recruiting costs and higher salaries for developers with those skills. Frameworks and programming languages heavily influence the architecture of your application, which affects the ability to scale cost-effectively.

What skills does your team have?

Think about what strengths your team has and how comfortable they are with infrastructure, coding, and cloud computing. If the team has minimal experience with infrastructure, you'll need to keep ease of use in mind as you evaluate providers and technologies.

Do you have both frontend and backend developers on the team?

What programming languages are they most comfortable with?

Do you have specific needs that require raw servers, or could a managed system provide for your needs?

Do you want to spend significant time maintaining infrastructure, or would you prefer managed resources to keep a lean team and give you more time to develop your application?

What is your ideal time to market?

What's the ideal timeline between proof of concept and release?

Do you have stakeholders counting on you to maintain a certain timeline?

If you have a growth plan, what will it take to accomplish that goal?

How scalable do you need the application to be, and how fast does that scaling need to happen?

Answering these questions will help you prioritize nimbleness and scalability as you consider providers. Making major changes to your tech stack down the road can add considerable costs or slow down development. Do your best to account for the future state of your application while maintaining simplicity for now.

What are your non-negotiables?

After considering everything you need, decide what's most important to you. Write down your non-negotiables and prioritize technologies that can accommodate what's most important for you and your growing business.

Servers and cloud computing

Infrastructure: Your team can use it to grow quickly.

Planning the infrastructure for your project begins with the hardware needed to power your computing needs. Historically, individuals would purchase servers and maintain their own hardware and software in a physical data center. This option is expensive to implement because the purchase cost of the hardware can be significant, and it can be costly to maintain because it requires individuals with technical expertise to upkeep the physical infrastructure. Scaling can also be complex, as you're limited by the capabilities of your on-premises hardware.

Because of the difficulties and costs associated with maintaining physical hardware, many businesses now choose an **Infrastructure as a Service (IaaS)** provider to deliver computing resources over the internet, including networking, storage, and other infrastructural components. This eliminates the need to maintain the physical hardware yourself and allows more flexibility as you scale and grow. Most businesses starting out use an IaaS model because of its ease of use, scalability, and lower cost.

Hack the Box scaled to meet demand while maintaining high performance with DigitalOcean. Read their story.

Providers like DigitalOcean, AWS, Microsoft Azure, and Google Cloud allow you to control your infrastructure without needing to control or manage the physical hardware. With these options, builders can set up their infrastructure and decide how involved they want to be with the upkeep. Cloud computing providers often offer a variety of products and services that make maintaining infrastructure easier, so you can decide how hands-on you want to be.

When exploring providers, it's essential to keep in mind your needs now and in the future. Avoid overspending on infrastructure and keep things simple as you get started. Make sure they have the flexibility that will allow you to scale and offer managed services that you may need down the road. Consider your integration needs and make sure the provider you choose can support you. With cloud providers, the price often increases with usage. Choose a provider that is transparent about its costs and offers a price point that is manageable in your early days and won't jump unexpectedly as you grow.

Operating systems

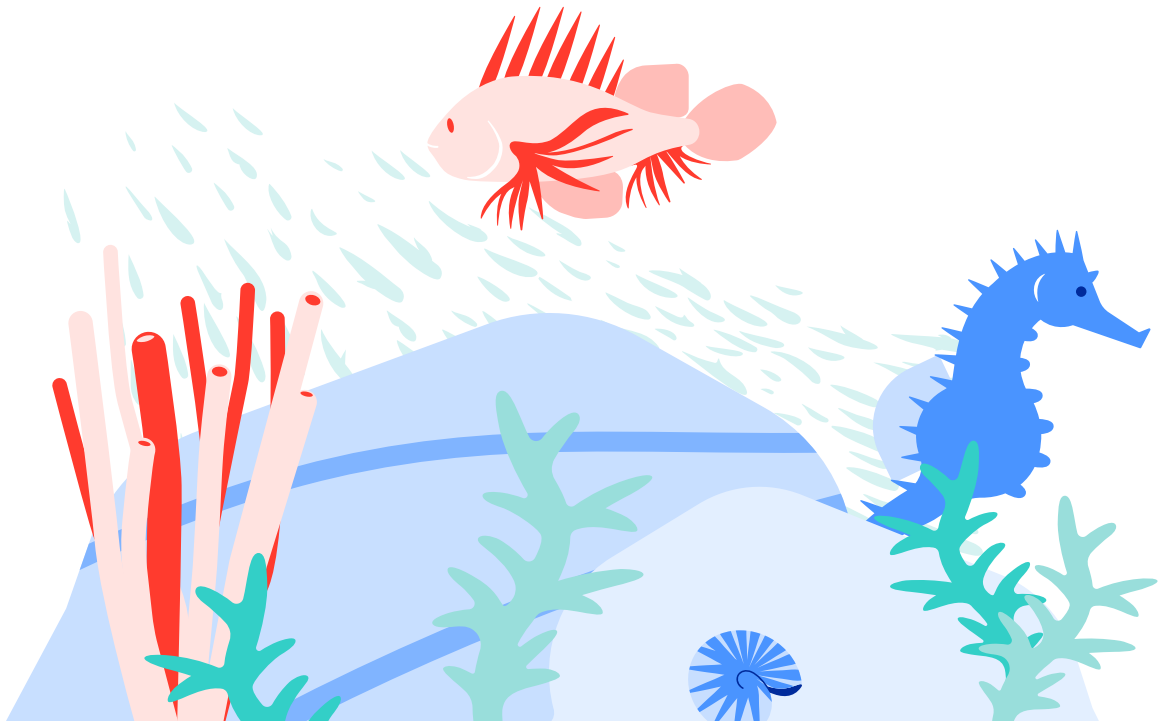
Versatility is key, but consider security and available support, too.

The operating system (OS) manages the computer's memory and processes and allows you to interact with the compute function.

It links together the hardware with the programs running on the computer. When considering which OS to use, consider each one's versatility, security, and cost, as well as what's most familiar and easy for you to use. Each OS will have an ideal use case, so keep your use case and experience in mind as you research your options. We recommend starting your search with Linux.

Linux-based OS are arguably some of the most popular OS choices. Linux is an open-source OS with significant support and a robust community. Its low cost and broad availability make it a popular choice for developers and SMBs. As you dig into the Linux ecosystem, you'll find that there are many Linux distributions—OS-based on a Linux foundation packaged with components from various projects. Linux or Linux distributions are often the OS of choice for cloud computing.

One extremely popular Linux distribution is Ubuntu. This open-source OS is well liked because of its low cost, ease of use, and enhanced security. Ubuntu is well supported, allowing new users to access a large community if they have questions or need help. Debian is another Linux distribution that's very popular and is best known for its stability. Many Linux distributions, including Ubuntu, are Debian-based.



Programming languages

How you build it is almost as important as what you build.

Programming languages are responsible for executing your application code, communicating with the database, and more.

Programming languages typically fall into three categories: **procedural, functional, or object-oriented**.

Procedural languages follow a set of commands in order.

Functional programming languages are based on applying sequential functions to solve complex problems.

Object-oriented programming languages are built on the concept of objects that contain both data and code to modify the data. Each language has distinct qualities and features, and which programming language you choose will depend heavily on the type of application you're building and your team's experience.

Tips for choosing a programming language

As a startup, you'll need to weigh your needs now and your potential needs in the future. Narrow down your choices by considering what will best suit the project itself—for example, only consider languages best suited for mobile development if you're building a mobile application, but then consider the following:

- What will the business ecosystem look like? What does your application need to be compatible or integrate with?
- If you need to hire additional developers, will they be available?
- How big is the community to offer support should you need it? Is there a vendor for the language?
- Think about the specific requirements for things like libraries, features, and tools.
- Are there any security considerations?

Common programming languages

There are many languages to choose from, each with different benefits, drawbacks, and levels of support. A few of the most common languages are:

Python. Python is an increasingly popular scripting language because it's beginner-friendly, easy to learn, and has a wide range of uses, including data science and visualization, machine learning, interfacing with databases, building web applications, and more.

Javascript. Javascript is another one of the most popular programming languages and can be used both for front-end development and server-side development. Javascript is relatively beginner-friendly, and there are many resources available to help developers learn.

Java. Java is different from JavaScript. Java is a general-purpose, object-oriented, high-level language. It's useful for web and application development but one of the more difficult languages to learn and use because its code is lengthy and complex. Java has full multiprocessing support and is the language of choice for enterprises. It's also popular in education and on Android if you're making a native application. Java's huge community of developers available to provide support and its stability and security keep it at the top of many people's lists.

PHP. PHP is a server-side language primarily used for web development, but its versatility allows it to be used with other programming languages fairly easily. PHP is very popular in open-source web applications from the last 10+ years, and much of the web is PHP.

Ruby. Ruby is an open-source object-oriented scripting language primarily used for web development. When it comes to building web applications, Ruby is easy to use and fairly easy to integrate with front-end frameworks. Ruby combined with Rails is extremely efficient and can save developers quite a bit of time. Ruby lacks the support that some of the other languages have.

C++. C++ is a procedural language that supports object-oriented and functional principles. It's a powerful general-purpose language primarily used in game development. C++ is hard to learn, with a very strict syntax that means a small mistake can cause a lot of errors. It's most useful in cases where you need very large, complex, monolithic desktop applications and wouldn't be the best choice for web applications.

Rust. Rust is syntactically similar to C++ and is gaining popularity because it's comparatively fast and easier to use. Rust has some enhanced safety features that maintain boundaries and preserve system integrity, helping the language avoid some of the pitfalls of C++.

Go. Go or Golang is a programming language that's known for its power and simplicity. It's particularly popular in infrastructure and cloud services, and its power and performance combined with ease of use have made it popular with many developers.

Databases and database management systems

Collect, store, and utilize your company data efficiently.

A database is simply a structured collection of data. While that data can have many uses, you need a database management system (DBMS) to access, interpret, and manipulate the data for your needs.

The DBMS will allow system administrators to track changes, pinpoint errors, and implement backup and recovery systems in addition to managing the data.

When considering which DBMS to use, start with the type of data you're collecting and using. Is it numeric or customer information like names, addresses, and more? Will you have multimedia that you're collecting and storing? Think about where you want the data stored, how you plan to manage it, and the structure that it needs to have to fit your needs. Also, keep in mind how you'll be using the data and what querying mechanisms you need.

Relational DBMS vs. NoSQL DBMS

Relational DBMSs are the most popular types of DBMS. A relational database stores its data in one or more tables consisting of rows and columns. Each column of a table represents an attribute and each row represents a record. Each field in a table represents a data value. Structured Query Language (SQL) is the language used to query relational database management systems, so relational databases are also commonly called SQL databases.

Common relational databases are Oracle, MySQL, MariaDB, MicrosoftSQL Server, and PostgreSQL. Consider using relational databases when you have simple, predictable, and structured data like financial information. Relational databases are easy to get up and running and do well when working with complex queries and running reports. Because of the relational nature of the database, it's more predictable to know where parts of the data live.

NoSQL databases are databases that do not use SQL as their primary language. NoSQL databases include several different subsets, defined by the way they store and interact with their data. Graph databases, network databases, object databases, and document databases are examples of NoSQL databases. Common NoSQL database management systems are MongoDB and Redis.

NoSQL databases offer more flexibility than SQL Databases, allowing users to leverage multiple data storage schemas like JSON files, lists, binary values, and complex networks of related information stored in graph formats. NoSQL databases are often faster than SQL databases. NoSQL databases can manage large volumes of data without sacrificing performance, which means applications relying on large datasets can quickly query them.

NoSQL databases can scale horizontally, enabling users to host databases across multiple servers and locations inexpensively, rather than scaling vertically through more powerful and expensive servers. They also have some developer-friendly features, like supporting agile development workflows, and are intended to make developers more efficient by enabling them to query data through simple code.

It's not uncommon to use both SQL and NoSQL databases together for the same use case to take advantage of the benefits of each.

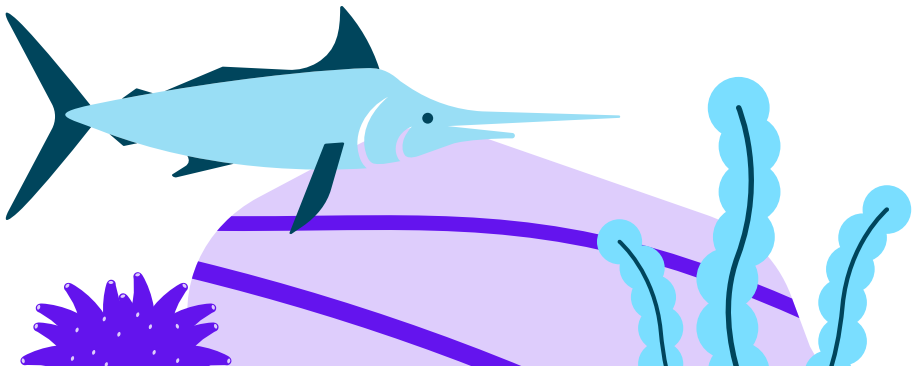
Performance monitoring services

Make sure your software is working as intended.

You'll likely want to have the ability to monitor the performance of your software to ensure that it's running smoothly and that your users have an optimal experience. Tracking things like application load and response times, CPU utilization, and consistent error documentation will help you provide the best experience for your users.

Common performance monitoring tools are:

- Server & Application Monitor
- New Relic APM
- SolarWinds AppOptics
- AppDynamics
- ManageEngine
- Datadog
- Prometheus
- Grafana



Standard tech stacks

Use common tech stacks to go easy on your developer teams.

The popularity of specific combinations of software has created what's considered standard tech stacks. Choosing one of these stacks allows you to access a community of developers using similar setups, and developers are often familiar with them and their processes, making it easier to find and hire developers for your project.

Two of the most commonly used stacks are MEAN, MERN, or MEVN, and LAMP, LNMP, or LLMP. The MEAN tech stack is a popular choice for web applications because apps can be written in one language for both client and server-side projects. LAMP is popular for developing simple web applications and dynamic websites.

MEAN, MERN, or MEVN

- MongoDBData
- Express.js
- Angular, React.js, or Vue.js
- Node.js

LAMP, LNMP, or LLMP

- Linux
- Apache, Lighttpd, or Nginx
- MySQL
- Perl, PHP, or Python

For native iOS and Android mobile application development, developers often use Swift language, Apple Xcode toolkit, and iOS SDK for iOS and Java or Kotlin languages, Android Studio, Android Developer Tools, and Android SDK for Android devices.

How others have done it*	airbnb	Pinterest	Uber
Programming Languages	JavaScript, Ruby	Python, Java, Go	Python, Java, Go, Objective-C
Framework	Rails	Django, Javascript MVC	Node.js, Apache Thrift
Databases	MySQL, Amazon RDS, Hadoop	MySQL, Hadoop, HBase, Memcached, Redis	MySQL, PostgreSQL, MongoDB, Redis
Server	NGINX	NGINX	NGINX

*Information via https://stackshare.io/stacks

Managed vs. self-managed tech

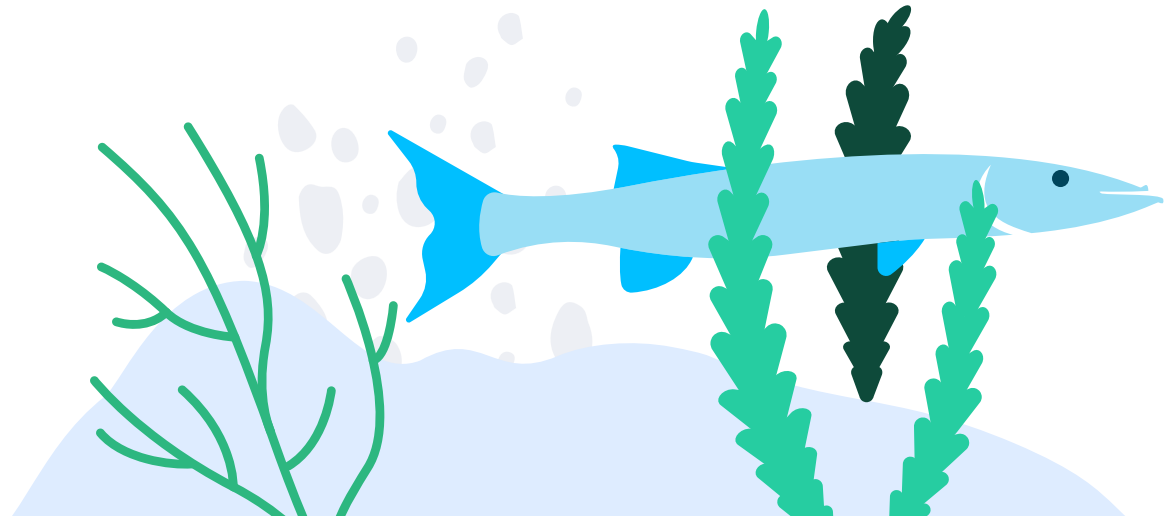
DIY or pay an expert—the right choice depends on your unique needs.

Many of the technologies you choose for your tech stack can be self-managed. If you have the time and the expertise to maintain your entire infrastructure internally, then self-managing gives you more insight and control than you would have with a managed solution. However, if your team doesn't have the extensive experience often needed to maintain these technologies, or if you'd like to save time and focus on developing your application, you can choose a managed implementation option.

Selecting a managed implementation option means paying a provider to create and manage that part of your tech. While there is a cost for managed services, the benefits are often worth it. For example, choosing a managed database option means that users don't have to set up or maintain a database independently and instead outsource the responsibility to oversee the database's infrastructure to the database provider. Databases need to be provisioned, configured, and continually maintained, taking time and expertise that many developers don't have. A provider-managed database takes all of that upkeep out of the developers' hands, saving time and relieving stress.

When choosing providers, consider your needs both now and in the future. While there may be inherent differences in how these tools are supported, look for a robust community or a great support team that will be available should you need help. When researching cloud providers, make sure they provide versatility in their services. Can you get both managed and unmanaged products? Can you easily integrate with other systems and software? Does the provider have a strong record of partnerships across the industry? A provider that offers a variety of solutions and integrations is crucial as you seek solutions for your needs.

Find out how DigitalOcean's managed Kubernetes helped Atom Learning maintain 50% growth month to month.



DigitalOcean

About DigitalOcean

DigitalOcean simplifies cloud computing so developers and businesses can spend more time building software that changes the world. With its mission-critical infrastructure and fully managed offerings, DigitalOcean helps developers, startups, and small- and medium-sized businesses (SMBs) rapidly build, deploy, and scale applications to accelerate innovation and increase productivity and agility. DigitalOcean combines the power of simplicity, community, open source, and customer support so customers can spend less time managing their infrastructure and more time building innovative applications that drive business growth.

> To get started, **sign up for an account at DigitalOcean.com.** For more information or help migrating your infrastructure to DigitalOcean, **speak to a sales representative.**