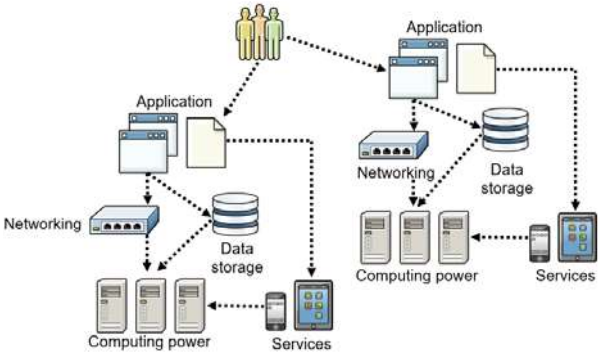
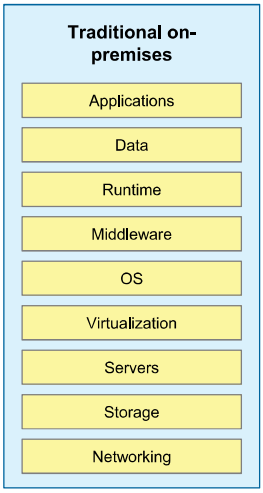
IBM Training

Before cloud computing





Introduction to cloud computing

Figure 1-4. Before cloud computing

Before you learn about cloud computing, you should know more about the software industry *before* cloud computing.

Before cloud computing, when you created a basic website for your clients, you started by developing your application with a programming language, such as Java, Node.js, or PHP. Then, you deployed it on a physical machine (server). On this server, you had an operating system and set up the configurations and middleware that were needed to run your application. Also, you needed a run time to run your application, such as Apache Tomcat or IBM WebSphere Liberty application server (if you used Java). Your application had to be linked to a database.

Then, to expose this application to your client, you needed an IP and domain name, and handled the network configurations, the physical location for your servers, and the electricity that was required for your servers. Security had to be set up and maintained. You had to manage the upgrades for these resources. You needed a large team of experts to install, configure, test, run, secure, and update these resources to keep your website running.

IBM Training

Challenges faced before cloud computing

- Cost
- Scalability
- Reliability
- Security
- Mobility

Figure 1-5. Challenges faced before cloud computing

Here some of the challenges that were faced before cloud computing:

Cost

It was costly to build reliable and maintainable software. You likely built you own infrastructure that might be needed to be turned into a whole data center, which included servers, network equipment, data storage, and so on. You needed to hire a team of experts to handle all these resources. All these factors made it difficult for small and midsized business to develop their own software and keep updating it.

Scalability

If there was high demand from your clients for your application, you needed to scale up the capacity of your application, which required more resources and some downtime to integrate and upgrade those resources. If demand decreased, you had some resources that were not used effectively.

Reliability

With any operation that is done to your server, such as maintenance and updates, you need downtime to perform those operations. Some issues that might have caused downtime for your application are power outages, hardware problems, general network issues, or even natural disasters.

Security

Security is necessary for all levels: application, network, infrastructure, and physical resources.

Mobility

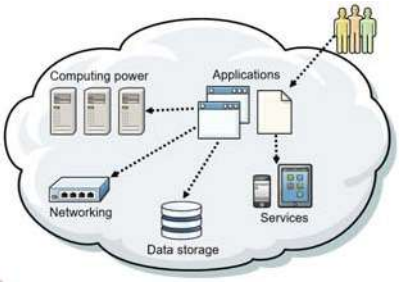
Part of the team worked onsite to, at least, set up the infrastructure and configured the network.

IBM Training

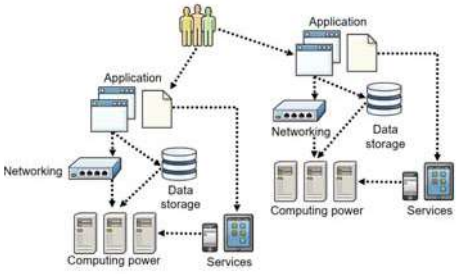
What is cloud computing

Cloud computing, often referred to as “the cloud,” is the delivery of on-demand computing resources (applications to data centers) on a pay-as-you-go basis.

- **Elastic resources**
- **Metered services**
- **Self-service**



Cloud computing model



Traditional on-premises computing model

Introduction to cloud computing

Figure 1-6. What is cloud computing

Cloud computing is a model for enabling convenient, on-demand access to provider-managed suite of both hardware and software resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Cloud computing is a disruptive change in the IT industry that represents a new model for the IT infrastructure that is different from traditional IT computing models. Cloud computing enables ubiquitous computing, where computing is available anytime and everywhere, using any device, in any location, and in any format. The surge of mobile devices is greatly contributing to this model.

This new model demands a dynamic and responsive IT infrastructure due to short application lifecycles. To support this model, new development processes, application design, and development tools are required.

Elastic resources: Scale up or down quickly and easily to meet changing demand.

Metered services: Pay only for what you use.

Self-service: Find all the IT resources that you need by using self-service access.

IBM Training

Characteristics of the cloud

Cloud makes hardware and platform resources readily available and quick to configure. Cloud provides the following characteristics to developers:

- On-demand resources
- Self-service
- Ubiquitous access
- Resource pooling
- Rapid elasticity
- Measured service

Figure 1-7. Characteristics of the cloud

Modern applications must be delivered quickly. Developers are pressured to get their product to market as soon as possible. They want to get feedback quickly, and then iterate on the idea to make the product better and faster.

The cloud makes hardware resources readily available and quick to configure, which shortens the time that is required for developers to show a working version of their products. Also, cloud allows the reuse of the same resources for multiple successive projects, which is more cost-efficient.

Characteristics of the cloud:

- On-demand resources: Have it when you need it with no need for tiresome preparation, downloads, and installations.
- Self-service: A customer can provision resources themselves by accessing a self-service portal and requesting the resource that they want.
- Ubiquitous access: Access the cloud from anywhere just by using an internet connection and a cloud account (user name and password).
- Resource pooling: Pooling hardware resources and abstracting them so that when resources are not being used by one customer, instead of sitting idle, those resources can be used by another customer.

- Rapid elasticity: Scaling up or down resource consumption is available on demand with any quantity and at any time.
- Measured service: Pay only for what you use, which helps you monitor any wastage of resources.

Examples of cloud resources include:

- Servers
- Storage
- Networks
- Security
- Applications
- Platforms
- Runtimes
- Databases
- Managed services

IBM Training

Benefits of the cloud

- Achieves economies of scale.
- Goes from CAPEX to OPEX.
- Runs anytime and anywhere.
- Facilitates the agile methodology.
- Ensures global availability.
- Built-in security
- Provides advanced capabilities.

Figure 1-8. Benefits of the cloud

Enterprises eager to undergo digital transformations and modernize their applications are quick to see the value of adopting a cloud-computing platform. They are increasingly finding business agility or cost savings by renting software and infrastructure. Each cloud-computing service and deployment model type provides you with different levels of control, flexibility, and management.

- Achieves economies of scale (do more with less).
  Economies of scale decrease costs because of increased production. These economies became achievable in software because of the flexibility of the cloud.
- Reduces capital expenditure (CAPEX) by moving to the operational expenditure (OPEX) model (use only when needed).
  CAPEX is the money that is used to acquire or update the assets of a firm. OPEX is the money that is used on running operations. So, in the software industry the “pay as you go” model helps you go from CAPEX to OPEX.
- Runs anytime and anywhere (access to services, on any device, and anywhere in the world).

- Facilitates agile methodology (faster time to market).
  Agile methodology is a development methodology where you engage the client with the development team and constantly get changing requirements that are embraced for the client's competitive advantage. Applying the agile methodology became achievable because of the cloud.
- Ensures Global Availability (focus on developing applications, and the rest automatically follows).
  It helps to improve reliability and provide a good disaster recovery plan with high availability.
- Built-in security

  Cloud providers typically have standards to build their environments and standardized practices to run operations that meet the security needs of enterprise clients. As a user of cloud, your application could benefit from higher orders of security by virtue of it being built into the cloud offering for all.
- Provides advanced capabilities (advanced technology that is readily available and you can experiment with).
  Many advanced technologies, such as big data and AI services that need high-computing-power capabilities would not be available without cloud computing.

IBM Training

Factors contributing to the growth of the cloud

- Applications with a short lead time to delivery.
- Developers expect to have programming language options and interact with predefined services.
- Modern applications must be able to scale and be managed dynamically.
- Developers expect the “pay-as-you-go” utility computing billing method.

Figure 1-9. Factors contributing to the growth of the cloud

One factor contributing to the growth of cloud computing is that today’s applications must be delivered quickly. Developers are pressured to get their product to market as soon as possible. They want to get feedback quickly, and then iterate on the idea to make the product better and faster.

The cloud makes hardware resources readily available and quick to configure, which shortens the time that is required for developers to show a working version of their products.

Another factor contributing to the growth of cloud computing is that developers expect to use many languages and interact with predefined services. Cloud computing provides prepackaged language support, which enables the support of many more languages than the traditional do-it-yourself environment. Cloud computing can also make available shared services that provide an externally managed way of delivering frequently used functions.

Another factor that drives the adoption of cloud computing is that developers want to be able to add resources to a specific application (*scaling up*, or *vertical scaling*), or add duplicate instances of an application (*scaling out*, or *horizontal scaling)* to handle increased customer load. Cloud platforms provide standardized methods to scale applications.

Developers expect the “pay-as-you-go utility” computing billing method that cloud provides.

1.2. Cloud service models

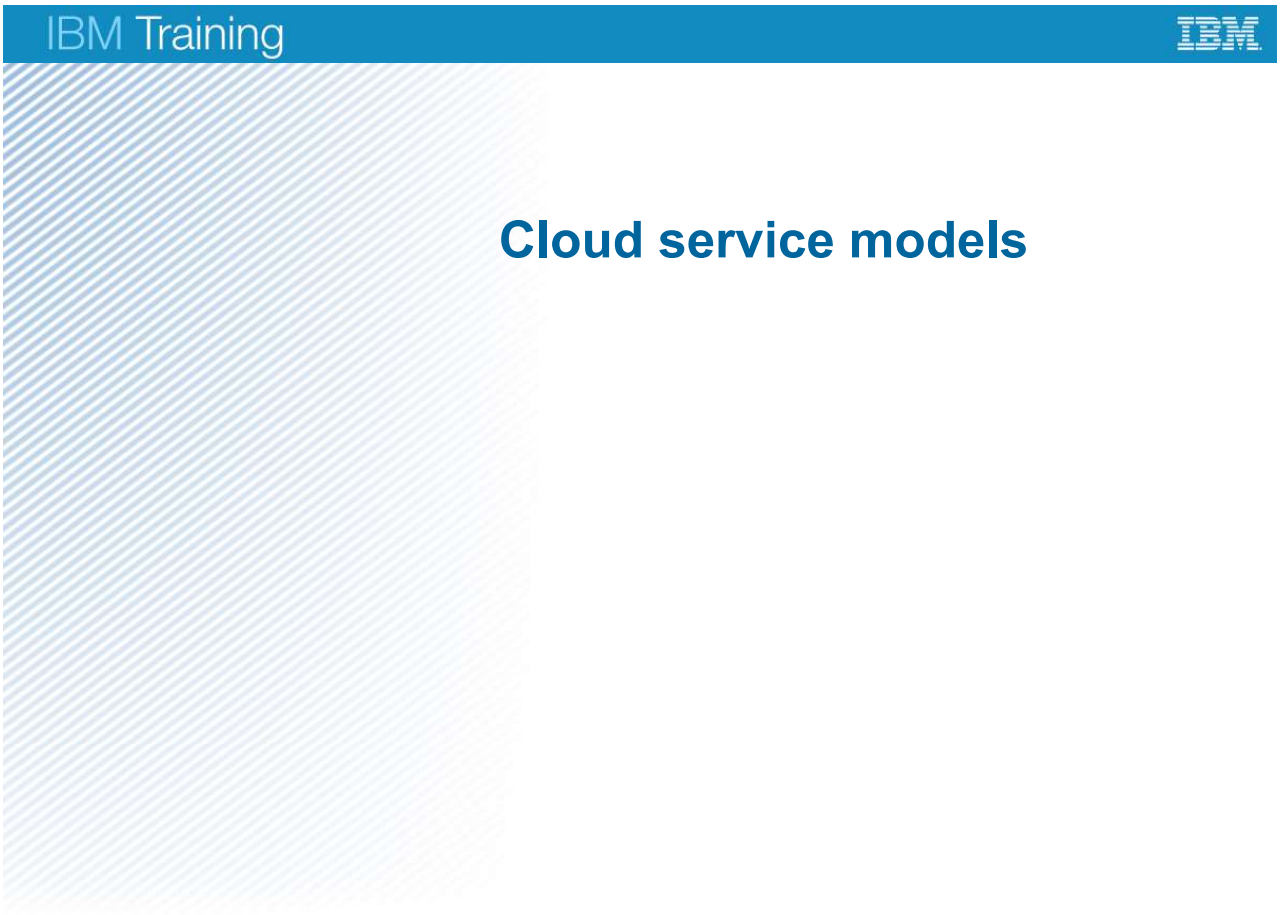


Figure 1-10. Cloud service models

IBM Training

Topics

- Introduction to cloud computing
- Cloud service models
- Cloud deployment models

Introduction to cloud computing

Figure 1-11. Topics

IBM Training

The pizza analogy

The cloud has different service models. With platform, infrastructure, and software offered as services, the pizza analogy is an easy way to understand this approach.

Traditional	Infrastructure as a Service (IaaS)	Platform as a Service (PaaS)	Software as a Service (SaaS)
Table & Chairs	Table & Chairs	Table & Chairs	Table & Chairs
Drinks	Drinks	Drinks	Drinks
Oven	Oven	Oven	Oven
Toppings	Toppings	Toppings	Toppings
Dough Base	Dough Base	Dough Base	Dough Base
Make from scratch at home	**Buy pizza and bake home**	**Get pizza delivered**	**Dine at Pizza Restaurant**
	= you furnish; = vendor furnishes		

Introduction to cloud computing

Figure 1-12. The pizza analogy

Cloud computing can be explained by using the pizza analogy:

- You build a pizza by preparing the dough, purchasing certain toppings, heating the oven, baking it, and then serving and eating the pizza along with drinks at home.
- *Infrastructure as a Service* (IaaS) is like buying a pre-made pizza from the supermarket. You bake it in your oven, serve it with drinks, and eat the pizza at home.
- *Platform as a Service* (PaaS) is like ordering a pizza from a pizza delivery restaurant. The pizza is prepared by the restaurant and delivered to your front door. You provide the drinks and eat it at home.
- *Software as a Service* (SaaS) is like going to a restaurant and eating the pizza there while enjoying the company of others and sharing the atmosphere of the restaurant.

When translating this analogy to the cloud, we can say that:

- To build an application, you must provide the infrastructure, platforms, operating systems, networking components, and so on, which is like making a pizza at home.
- With IaaS, you order hardware and an infrastructure. Often, this infrastructure is managed for you. You deploy only the middleware, runtime, and your application. The infrastructure is like the pizza that is pre-made, and you bake it to your liking.
- PaaS is like getting a pizza delivered. The pizza is ready to be eaten, and need only to provide drinks to go with it. In the cloud, this means that the cloud provider offers access to the platform and runtime and you only need to push the application.
- SaaS is using an application that is hosted at the cloud provider, which is similar to going to a restaurant and enjoying your pizza there.

IBM Training

Cloud service models

IaaS: Infrastructure as a Service

PaaS: Platform as a Service

SaaS: Software as a Service

Figure 1-13. Cloud service models

IaaS

A cloud provider offers clients pay-as-you-go access to storage, networking, servers, and other computing resources in the cloud.

PaaS

A cloud provider offers access to a cloud-based development environment in which users can build and deliver applications. The provider supplies and manages the underlying infrastructure.

SaaS

A cloud provider delivers software and applications through the internet that are ready to be consumed. Users subscribe to the software and access it through the web or vendor application programming interfaces (APIs).

IBM Training

Infrastructure as a Service

Key features:

- Instead of purchasing hardware, users pay for IaaS on demand.
- Infrastructure is scalable depending on your processing and storage needs.
- You avoid the cost of buying and maintaining your own hardware.
- Enables the virtualization of administrative tasks, which frees time for other work.

Figure 1-14. Infrastructure as a Service

IaaS is a cloud computing offering in which a vendor provides users access to computing resources, such as servers, storage, and networking.

IaaS offerings are built on top of a standardized, secure, and scalable infrastructure. The virtualization of the hardware is performed by a program that is known as a hypervisor. A hypervisor manages *virtual machines (VMs)* or *virtual servers*, which hosts multiple operating system instances that are running on a specific physical machine. Each operating system appears to have the host's processor, memory, and other resources all to itself, but in reality the hypervisor is controlling and provisioning access.

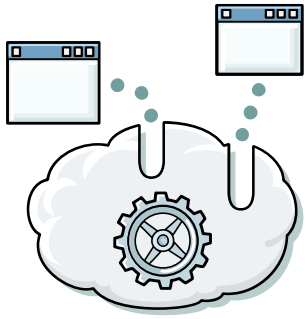
Organizations use their own platforms and applications within a service provider's infrastructure.

IBM Training

Platform as a Service

Key features:

- PaaS provides a platform with tools to test, develop, and host applications in the same environment.
- Enables organizations to focus on software development without having to worry about the underlying infrastructure.
- Providers manage security, operating systems, server software, and backups.
- Facilitates collaborative work even if teams work remotely.



Introduction to cloud computing

Figure 1-15. Platform as a Service

PaaS is a cloud-computing offering that provides users with a cloud environment in which they can develop, manage, and deliver applications.

In addition to instances and other computing resources, users can use a suite of prebuilt tools and run times to develop, customize, and test their own applications.

PaaS typically entails the developer uploading the application code, or pointing to it and letting the PaaS complete the following tasks:

1. Obtain the runtime binary files and dependencies for the application.
2. Structure their application code into the correct directory tree for containerization.
3. Provision a container (or set of containers) on which the application can run.
4. Automatically generate a simple and basic networking configuration for access to the application.
5. Provide automatic and built-in monitoring of the application.
6. You can update and redeploy the application with zero downtime.

PaaS typically involves sacrificing some level of fine-grained control over the application's environment to gain convenience, ease of use, and rapid deployment by using a predefined deployment process. PaaS also uses external services or APIs that allow rapid composition of applications by reusing pieces of infrastructure (for example, a database) that require little to no investment in setup and configuration.

PaaS also gives the developer an automatic method for scaling. For example, consider a situation where the developer wants more hardware resources that are dedicated to an application (scaling up or vertical scaling) or more instances of the application to handle the load (scaling out or horizontal scaling). PaaS also provides built-in application monitoring. For example, the platform sends notifications to inform developers when their application crashes.

IBM Training

Software as a Service

Key features:

- SaaS vendors provide users with software and applications through a subscription model.
- Users do not have to manage, install, or upgrade software; SaaS providers manage all of those items.
- Data is secure in the cloud; equipment failure does not result in loss of data.
- Applications are accessible from almost any internet-connected device from anywhere in the world.

Introduction to cloud computing

Figure 1-16. Software as a Service

SaaS is a cloud-computing offering that provides users with access to a vendor's cloud-based software. Users do not install applications on their local devices.

Instead, the applications are on a remote cloud network that is accessed through the web or an API. Through the application, users can store and analyze data and collaborate on projects.

IBM Training

Example of cloud services

IaaS

- Virtual servers
- Bare metal machines
- Block storage
- File share storage
- Object storage
- Backup
- IP management
- Virtual private networks
- Firewalls
- Load balancers
- Automation

PaaS

- Run times and development platforms
- Databases
- Analytics
- Integration
- Starter kits
- Mobile platforms
- Push notifications
- Messaging
- Developer tools
- Continuous integration / continuous delivery

SaaS

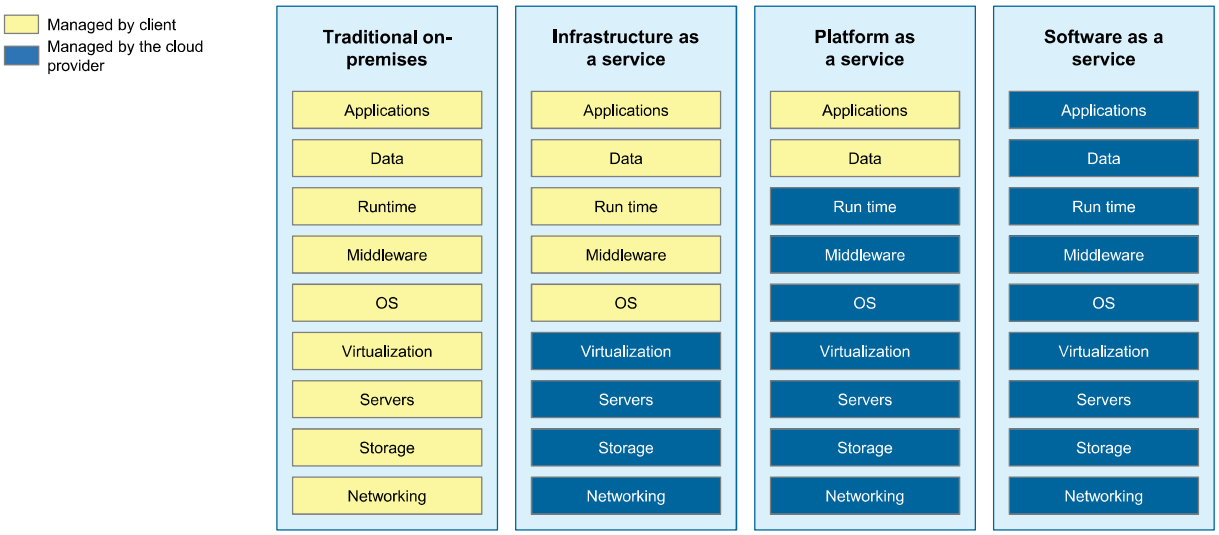
- Email and Collaboration
- Customer relationship manager (CRM)
- Enterprise resource planning (ERP)
- Video streaming
- Marketing
- Talent management
- Advertising

Figure 1-17. Example of cloud services

This slide shows examples of the services that are available for each model.

IBM Training

Cloud provider and client responsibilities



Introduction to cloud computing

Figure 1-18. Cloud provider and client responsibilities

This slide shows the split between the provider and client responsibilities when dealing with on-premises or “as a service” scenarios.

Typically, the cost decreases as you move to the right in the scenarios that are shown in the slide; however, the flexibility also is reduced.

Organizations or departments within an organization make their own cost-based decisions about which delivery model to use for individual applications or projects.

Most enterprises end up using some combination of all of the models that are shown in the slide.

IBM Training

Choices when building cloud applications

When developing applications for the cloud, developers have many options to choose from in terms of platforms, frameworks, tools, and services:

1. Traditional development
   - Example: Bare metal or VMs
2. Containerization
   - Example: Docker or Kubernetes
3. PaaS
   - Example: Cloud Foundry
4. Serverless and functions
   - Example: OpenWhisk

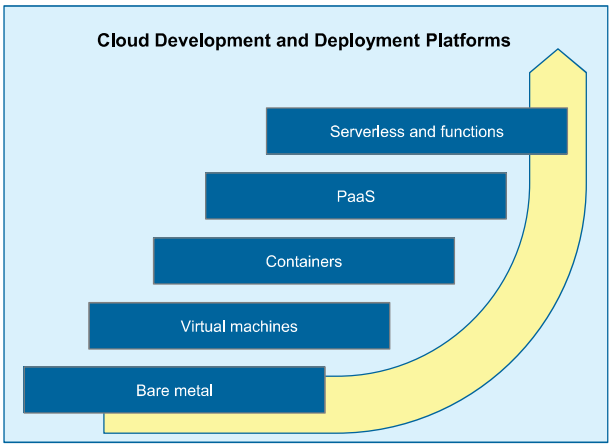


Figure 1-19. Choices when building cloud applications

Developers have many options when deploying their applications to the cloud:

1. Traditional development involves deploying applications to bare metal servers or VMs. Bare metal servers enable isolation and single tenancy, and VMs enable shared resources and multi-tenancy. When using traditional deployment, the developers are responsible for deploying and managing all the needed runtimes and dependencies for the application.
2. With containers, developers can package applications with runtimes by using a lightweight packaging medium that is called *containers*. Unlike VMs, which provide hardware virtualization, containers provide operating system-level virtualization.
3. PaaS solutions such as Cloud Foundry enable developers to focus on coding and pushing code to a platform that takes care of deployment and scaling. Developers do not need to worry about packaging or maintaining the host that runs the applications.
4. Server-less is a way for developers to focus on the development of event-based applications. The application code is broken into separate functions, and each function can be independently triggered by an event or an action, such as an API call.

1.3. Cloud deployment models

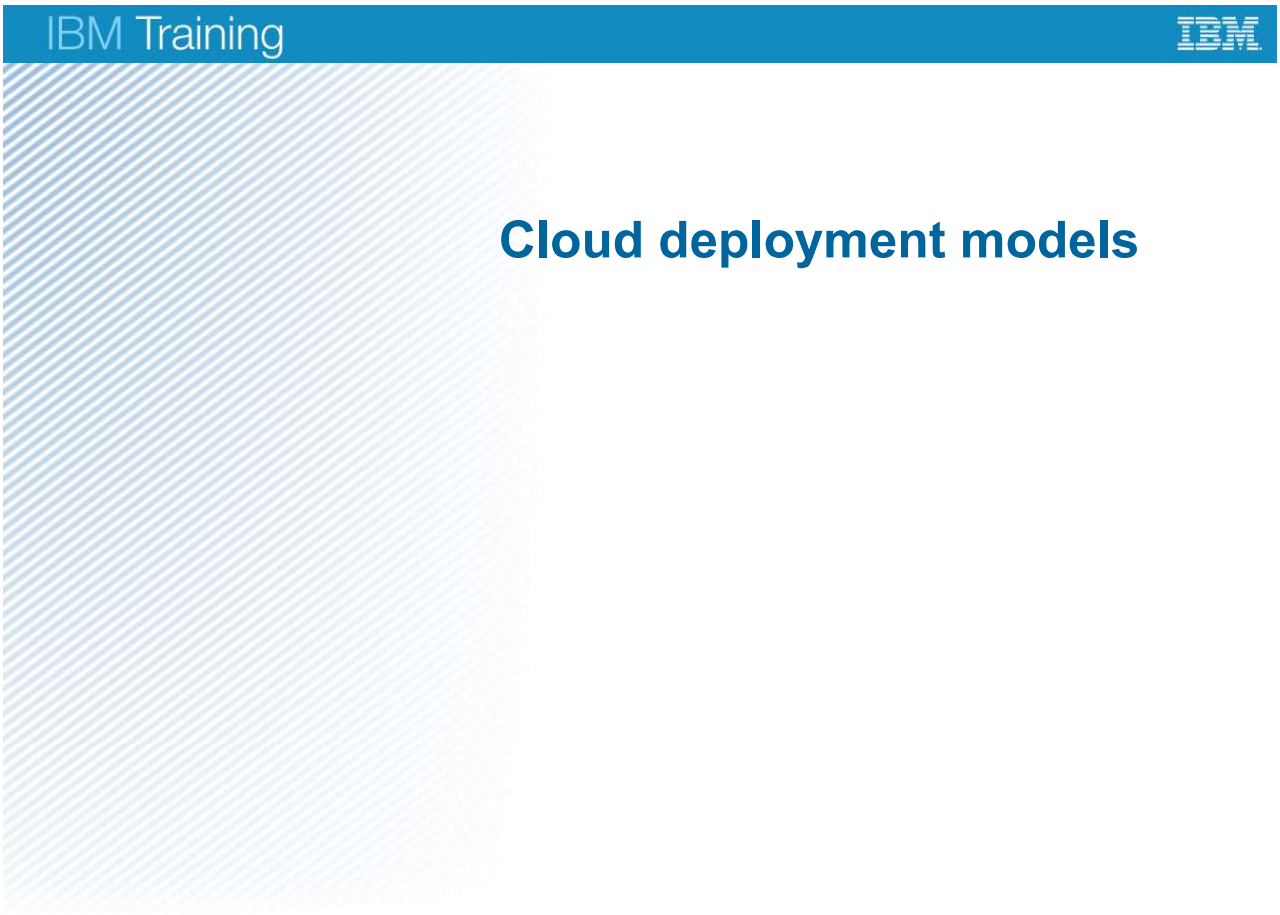


Figure 1-20. Cloud deployment models

IBM Training

Topics

- Introduction to cloud computing
- Cloud service models
- Cloud deployment models

Introduction to cloud computing

Figure 1-21. Topics

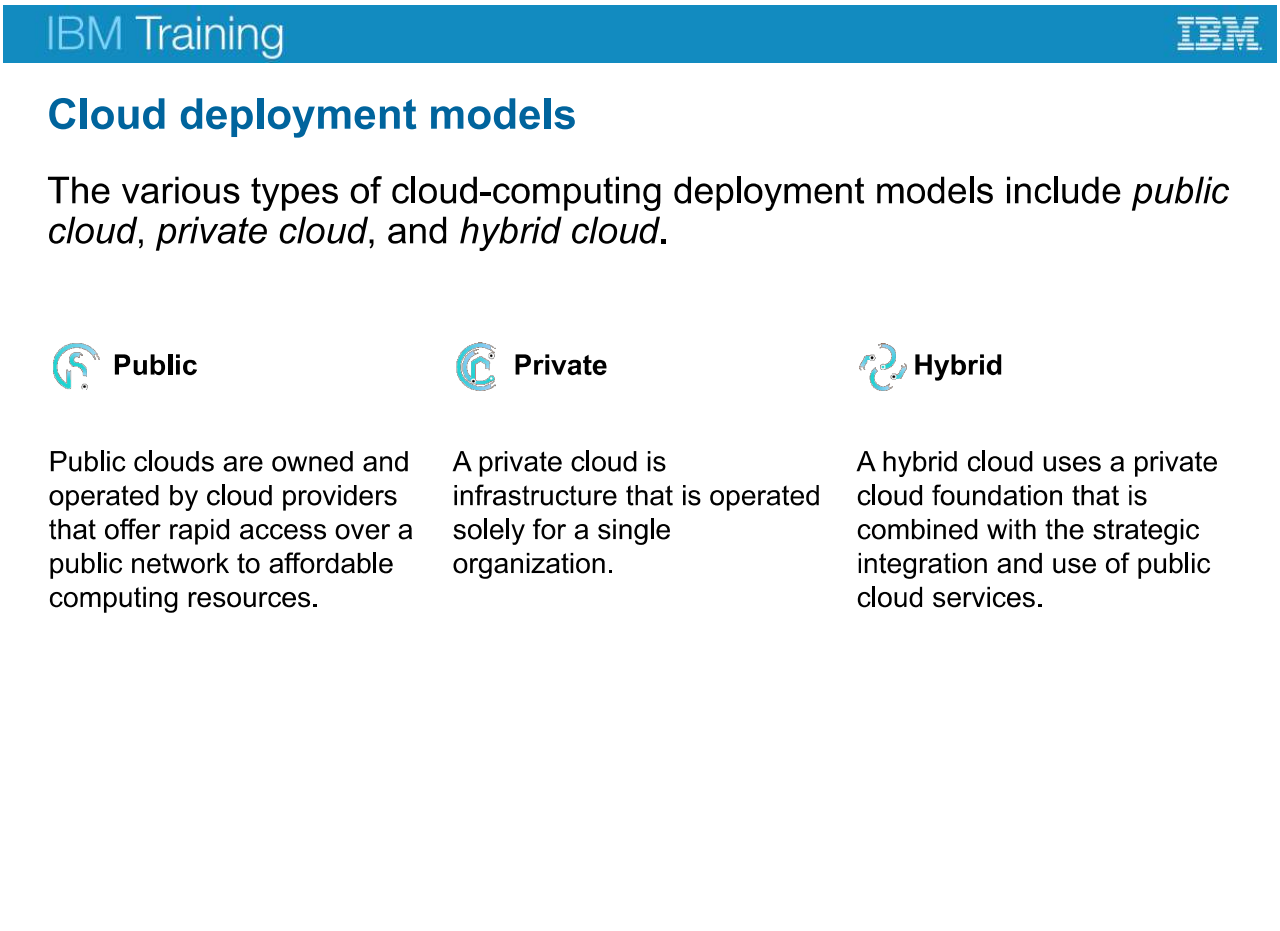


Figure 1-22. Cloud deployment models

Public cloud

Public clouds are owned and operated by cloud providers that offer rapid access over a public network to affordable computing resources.

Here are key aspects of a public cloud:

- Enables flexible and scalable IaaS for storage and compute services at a moment's notice.
- Enables powerful PaaS for cloud-based application development and deployment environments.
- Gives access to innovative SaaS business apps for applications ranging from customer resource management (CRM) to transaction management and data analytics.

Private cloud

A private cloud is an infrastructure that is operated solely for a single organization. Private clouds can take advantage of cloud's efficiencies while providing more control of resources and allowing clients to steer clear of multitenancy.

Here are key aspects of a private cloud:

- Provides self-service interface controls services, which enable IT staff to provision, allocate, and deliver quickly on-demand IT resources.
- Facilitates highly automated management of resource pools for everything from compute capability to storage, analytics, and middleware.
- Provides sophisticated security and governance for a company's specific requirements.

Hybrid cloud

A hybrid cloud uses a private cloud foundation that is combined with the strategic integration and use of public cloud services. Most companies with private clouds evolve to manage workloads across data centers, private clouds, and public clouds, which creates hybrid clouds.

Here are key aspects of a hybrid cloud:

- Enables companies to keep critical applications and sensitive data within a traditional data center environment or private cloud.
- Enables taking advantage of public cloud resources like SaaS for the latest applications and IaaS for elastic virtual resources.
- Facilitates portability of data, apps, and services and more choices for deployment models.

IBM Training

Cloud-native applications

Cloud-native applications capitalize on the scalability and flexibility of the cloud:

- Applications are broken into separate services called *microservices*.
- Microservices can be developed in different programming languages (polyglot development).
- Microservices communicate with each other by using an agreed upon protocol (such as REST or gRPC).
- Microservices work together as a whole to make up an application, yet each can be independently scaled, continuously improved, and quickly iterated through automation and orchestration processes.

Figure 1-23. Cloud-native applications

Here are some advantages of cloud-native apps:

- Compared to traditional monolithic apps, cloud-native applications can be easier to manage as iterative improvements occur through agile and DevOps processes.
- Composed of individual microservices, cloud-native applications can be improved incrementally and automatically to add continuously new and improved application features.
- Improvements can be made non-intrusively, causing no downtime or disruption of the user experience.
- Scaling up or down proves easier with the elastic infrastructure that underpins cloud-native apps.
- The cloud-native development process more closely matches the speed and innovation that is demanded by today's business environment.

Even with the advantages that are provided by cloud-native applications, there are also some disadvantages to consider:

- Although microservices enable an iterative approach to application improvement, they also create the necessity of managing more elements. Rather than one large application, it becomes necessary to manage far more small and discrete services.
- Cloud-native apps demand more toolsets to manage the DevOps pipeline, replace traditional monitoring structures, and control microservices architecture.
- Cloud-native applications allow for rapid development and deployment, but they also demand a business culture that can cope with the pace of that innovation.

Cloud-native versus cloud-enabled

A cloud-enabled application is an application that was developed for deployment in a traditional data center, but was later changed so that it also could run in a cloud environment. However, cloud-native applications are built to operate only in the cloud. Developers design cloud-native applications to be scalable, platform-neutral, and composed of microservices.

IBM Training

Cloud-native development methods

When developing cloud-native applications, developers must understand and adopt new methods and patterns to maximize the capability that is provided by the cloud provider:

- Readily available sandbox and production environments
- Programming languages and frameworks
- APIs
- Developer toolchains

Figure 1-24. Cloud-native development methods

When developing cloud-native applications, developers must understand and adopt new methods and patterns to maximize the capability that is provided by the cloud provider:

- Provides readily available sandbox and production environments. These environments offer the following capabilities that are attractive for developers:
  - Free trials that are offered with most products.
  - Pre-built templates and examples that help developers to get started fast.
  - Easier to understand the application lifecycle.
  - The environment to run an application is set up in minutes instead of days.
- Brings a wide range of choices to developers in the following areas:
  - Programming languages and frameworks.
  - Services.
  - APIs.
- A developer toolchain facilitates integrated development, test, and debugging:
  - The new model is to integrate development and operations teams into DevOps.
  - Build engine for compilation and testing.

IBM Training

Cloud-native development methods (cont.)

Cloud-native development introduces the 12-factor app methods and patterns to development:

I. Codebase: One codebase that is tracked in revision control, but there are many deployments.
II. Dependencies: Explicitly declare and isolate dependencies.
III. Configuration: Store the configuration in the environment.
IV. Backing services: Treat backing services as attached resources.
V. Build, release, and run: Strictly separate build and run stages.
VI. Processes: Run the app as one or more stateless processes.

Figure 1-25. Cloud-native development methods (cont.)

Examples for the first six factors:

- Factor 1. One codebase that is tracked in revision control with multiple deployments: Use one codebase for tracking and revision control, and from that codebase, you can deploy many times.
- Factor 2. Explicitly declare and isolate dependencies. For example, using a package.json file for a Node.js application lists all the external dependencies.
- Factor 3. Store configuration in the environment: Provide any extra configuration information as environment variable that can be bootstrapped when the application starts.
- Factor 4. Treat backing services as attached resources: Bind services to applications by using attach services (such as DBaaS or load balancers) to an application.
- Factor 5. Strictly separate build and run stages: For example, it is impossible to make changes to the code at run time, since there is no way to propagate those changes back to the build stage.
- Factor 6. Run the app as one or more stateless processes: When you design applications, use multiple processes or services as needed. Avoid dependencies on sticky sessions and keep session data in a persistent store to ensure traffic can be routed to other processes without service disruption.

IBM Training

Cloud-native development methods (cont.)

Cloud-native development introduces the 12-actor app methods and patterns to development:

VII. Port binding: Export services by using port binding.
VIII. Concurrency: Scale out by using the process model.
IX. Disposability: Maximize robustness with fast startup and graceful shutdown.
X. Dev/prod parity: Keep development, staging, and production similar.
XI. Logs: Treat logs as event streams.
XII. Admin processes: Run admin and management tasks as one-off processes.

Figure 1-26. Cloud-native development methods (cont.)

Examples of the last six factors:

- Factor 7. Export services by using port binding: Cloud providers offer processes or services with a port for binding and then handle routing traffic to the process over this port automatically. Application code reads the port from the environment and binds to this port.
- Factor 8. Scale out by using the process model: Horizontal scaling of application instances in cloud providers.
- Factor 9. Maximize robustness with fast startup and graceful shutdown: Use a disposable approach to the design of a process in the application. There should be minimal startup actions required. When a process is terminated, it should exit with minimal housekeeping, which improves robustness and responsiveness to horizontal scaling events.
- Factor 10. Keep development, staging, and production similar: This approach enables agile software delivery and continuous integration.
- Factor 11. Treat logs as event streams: Cloud providers have ways to accumulate log data across various components of the application to enable analyzing or exporting to a third-party logging service.

- Factor 12. Run administrator and management tasks as one-off processes: Design tasks that must run once or occasionally as separate components that can be run when needed instead of adding the code directly into another component. For example, if an application must migrate data into a database, place this task into a separate component instead of adding it to the main application code at startup.