Best practices for deploying your apps in the cloud

Grant Shipley

April 02, 2015

Explore the various types of cloud computing systems available, and best practices that can help you with real-world application deployments on top of a cloud infrastructure.

As a developer, you probably hear a lot about new technologies that promise to increase the speed at which you can develop software, as well as ones that can increase the resiliency of your applications once you have deployed them. Your challenge is to wade through these emerging technologies and determine which ones actually hold promise for the projects that you are currently working on. No doubt, you are aware that cloud computing offers great promise for developers. However, you might not know about the areas where this technology can provide value to you and your projects. You also may not know the best practices to employ when implementing a project in the cloud. This article explores the types of cloud computing systems available, and provides guidelines that can help you with real-world application deployments on top of a cloud infrastructure.

Choosing between IaaS, PaaS, and SaaS

When people begin discussing cloud computing they are generally speaking about one of three possible deployment choices for application code: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS). Which one is right for your project depends on your specific needs for the code base that you are working on. Let's examine each one of these cloud choices.

Infrastructure as a Service (IaaS)

IaaS is a platform where an infrastructure is provided for you. With a click of a button, you can spin up virtual machines hosted by a provider with an operating system of your choice. The vendor providing the machine is responsible for the connectivity and initial provisioning of the system, and you are responsible for everything else. The vendor provides a machine and an operating system, but you need to install all of the software packages, application runtimes/servers, and databases that your application requires. Generally, IaaS requires that you have a team of system administrators to manage the system and apply firewall rules, patches, and security errata on a frequent basis.

Pro: You have complete control over every aspect of the system.

Con: You need system administration knowledge or a team of system administrators to maintain the system(s), since you are responsible for their uptime and security.

Platform as a Service (PaaS)

PaaS is a fairly new technology stack that runs on top of IaaS and was created with the developer in mind. With the PaaS platform, everything is provided except the application code, users, and data. Typically, when using a PaaS, the vendor maintains the application server, databases, and all of the necessary operating system components, giving you time to focus on the application code. Since the vendor manages that platform for you, it is often hard to open up ports that are not specifically called for the application server, runtime, or database in use. PaaS also provides features that are specifically meant for applications, including the ability to scale the application tier up based upon the user demand of the application. In most platforms, this happens with little-to-no interaction from the developer.

Pro: PaaS provides a complete environment that is actively managed, letting you focus on your application code.

Con: Developers are often restricted to certain major/minor versions of packages available on the system so that the vendor can manage the platform effectively.

Software as a Service (SaaS)

With the SaaS platform, everything is provided for you except the users and the application data. The vendor provides the application code and the developer has limited access to modify the software in use. This is typically not a choice for deploying custom applications, as the vendor provides the entire software stack. Hosted web email clients and hosted sales automation software are two good examples of how SaaS is used.

Pro: The entire stack is provided by the vendor except the application users and the associated data.

Con: You have limited control over the hosted application and it's often hard to integrate external workflows into the system.

Which should you choose?

As an application developer, you should choose PaaS, given that the infrastructure is managed for you, so you can focus on your application code.

Application scaling

As mentioned previously, PaaS provides scaling out of the box for most languages and runtimes. However, as a developer you need to be aware of the types of scaling offered and when it makes sense to scale horizontally or vertically.

Vertical scaling

Vertical scaling refers to a type of scaling that has been the default choice for decades. This type of scaling refers to the notion that to handle load, you simply use larger systems. This is one of the reasons why there are servers in place today with a terabyte of RAM and a massive number of CPUs and cores to serve a single Java® application. Typically when using vertical scaling, a single large system is used to handle most or all of the application requests from the users.

Horizontal scaling

With horizontal scaling, the application load and requests are spread over a group of smaller servers that are typically behind a load balancer. As a request from a user is made, the load balancer sends the request to a server and then manages the session state across the cluster of servers. There are often two types of horizontal scaling that can be utilized to ensure the best possible experience for the users of your application, manual and automatic scaling.

Manual scaling

With manual scaling, you specify that you want the application to scale up to handle increased traffic when you know you have an upcoming event that will increase application demand. For example, if you know that you are going to be running a marketing campaign to attract more users to your application, you might want to proactively add additional servers to your cluster. Most PaaS providers allow you to accomplish this task with a simple command.

Automatic scaling

With automatic scaling, you specify conditions where your application will automatically scale without any human interaction. This condition can be based on such things as the number of concurrent HTTP requests your application is receiving, or the amount of CPU usage that your application is using. This enables the developer to automatically add new servers to the load balancer when the demand for the application is high. Automatic scaling provides a truly hands-off approach to scaling while ensuring that demand from the users is met in a timely fashion. Automatic scaling is crucial when you have unplanned usage of your application due to certain circumstances -- such as getting your mobile application featured on an application store for a short period of time where your back end services reside in the cloud.

Which should you choose?

As a developer, you should choose a platform that allows for **both** manual and automatic horizontal scaling of your application.

Application state

Most cloud providers that provide a PaaS want you to start with green field development -- meaning, projects that are not affected by the constraints of prior work. Porting existing or legacy applications to the platform can be a challenge, mainly because the file systems in place are ephemeral in nature and do not allow for saving application state or resources on the file system. This restriction is why you might hear that you need to think about future applications as being stateless. To receive the benefits of an infrastructure that resides in the cloud, you need to employ stateless application design in your projects. To achieve that, take into account the following practices for new applications:

- Allow the application server or container to maintain the session state of the user across the cluster instead of relying on the file system.
- Do not store files or user assets on the physical file system of the server that your code is deployed to. Instead, consider using a cloud-based storage service and delivering assets through the provided REST API for the storage service.
- Use a database for storing assets related to a user if you do not have access to use a cloud storage API.

Which should you choose?

For green field applications, you should design applications that are stateless, meaning they do not store user assets or resources on the file system. For legacy or existing applications, choose a PaaS provider that supports both stateful and stateless applications.

Databases for cloud-enabled applications

Almost all applications being created today rely on a database of some type on the back end to store and retrieve information to be presented to the user. When developing applications for the cloud, you must also take into consideration what databases you will be using and where those databases should be located. Should the database be hosted on the same server(s) as the application, or is it better to house the database on a separate server or container?

In many cases, an application relies on information that's stored in a database that resides behind a corporate firewall, while the application front end is deployed on the public cloud. When this is the case, you have a couple of options for effectively accessing the information that you'll need to present to the user on the front end.

Option 1: Choose a provider that allows you to open up a remote VPC connection back to your database.

Option 2: Communicate to the database through a set of authenticated REST services deployed on the infrastructure that have access to the data.

Both of these options have inherent security risks that you need to consider when connecting to a database behind a corporate firewall from an outside cloud application. When this is the case, your best option is to select a cloud PaaS vendor that allows you to deploy your applications on a non multi-tenant environment.

If your application code does not need to connect to an existing corporate database, the number of options that you have are almost endless. I suggest that you deploy your database in the same geography/datacenter/region as your application code but on different containers or servers than your front-end application code. Use this option to scale the database independently of the web tier. Also, be sure to choose a database that scales quickly and easily regardless of whether it's a SQL or NOSQL database.

Considerations for multiple geographies

One of the great benefits of cloud computing is that you can deploy your application infrastructure throughout the world with little or no up-front cost. For example, deploying an application that has servers in both North America and EMEA has traditionally incurred a huge up-front cost to purchase and provision hardware and data centers. With an infrastructure that resides in the cloud, you can effortlessly deploy your application across as many geographies as your vendor supports. For simple applications that only have a limited number of users, this is not required. However, having access to deploy code in multiple geographies is critical to winning customer satisfaction by locating the application code as close to your target audience as possible.

Throw in the ability to manually or automatically scale your application across different geographies, and you'll have a really interesting value proposition on your hands by incurring a lower cost than deploying a traditional IT infrastructure.

Which should you choose?

Choose a cloud provider that enables you to both deploy and scale your application infrastructure across multiple geographies throughout the world to ensure that your audience has a fast and responsive experience while using your application.

Create and use REST-based web services

As you can see, deploying your application code in the cloud provides many benefits -- and one crucial benefit for high-demand applications is the ability to scale out the web and database tiers independently. That being said, it is also good practice to separate your business logic into web services that your front-end code can consume. Use this practice to scale out the web services tier independently from both the database and the front-end code. Separating your application logic from the presentation tier opens new doors for technologies that you might not have considered in the past, such as creating a single-page application using a language like Node.

Continuous delivery and integration

DevOps seems to be the latest buzzword that is gaining a lot of attraction across enterprises. To get ahead, you should probably start looking at and implementing both continuous integration and delivery on your next software project. When deploying applications to a cloud-based infrastructure, make sure you have workflows in place on your existing build system so that code can be deployed across the different environments. Fortunately, most of the more popular build systems provide plugins for some of the top cloud providers today, making it easy to configure your deployment rules based upon the correct permissions of who has access to deploy code to each environment. If you are not currently using a build system for your development team, start using one now!

Which should you choose?

Choose a cloud provider that meets all of the requirements above with the added feature of integrated Continuous Integration/Continuous Delivery (CI/CD) tools on the platform. The provider you choose should allow you to deploy your own build system or have the ability to easily integrate with existing systems that reside outside of the cloud platform.

Avoid vendor lock-in

If you take one thing away from this article, I hope this is it: While many cloud providers provide great-looking proprietary APIs that reduce the amount of code or work that you have to do, you should avoid them at all costs. This is nothing more than a simple ploy to get you locked into their ecosystem while making it extremely hard to move your application to another provider or to your own data center running in-house. To avoid these custom APIs, stick with tried-and-true technology stacks across your application, including the database tier, storage tier, and any micro service endpoints that you might want to create. While the up-front investment can be a bit higher than using a proprietary solution out of the box, your technical debt is greatly reduced, which can save you money and time in the long run.

Developing locally or in the cloud

As developers, we often code applications on our local system and then, when we reach a work milestone, move our code to the team's development environment. Most developers wish they could develop on a daily basis with an infrastructure that resembles production as closely as possible. That can often be challenging due to the system administration overhead incurred to provide each developer with a cluster of machines. Now that PaaS is available, every developer should begin to develop and deploy their code in the cloud. Most integrated development environments (IDE) provide plugins to streamline the process and make it feel as close to developing locally as possible.

Which should you choose?

Choose an IDE that provides a plugin for the cloud provider of your choice. Consider choosing a provider that provides the ability to hot deploy application code as well as the ability to enable remote debugging of your source code. Once you have selected a provider that offers these two things, you can continue to set break points inside of your IDE and step through code just as if you were deploying locally. This enables you to more quickly catch bugs that only appear when moving to a clustered environment.

What to look for in the coming year(s) from cloud providers

This article focused on the current state of applications being deployed to the cloud. One thing to look for and consider this year and next is the mass-industry movement to container-based deployments -- you have probably already heard about Docker and rocket containers. When selecting a cloud provider, make sure the roadmap for application migration to container-based deployments is called out clearly with a timeline that clearly defines your migration path. Also, be on the lookout for vendors that are sticking with industry-standard solutions around containers and orchestration, versus creating proprietary solutions.

Conclusion

Cloud computing has many benefits that you should take advantage of in your daily software development and deployment to make your software more stable, scalable, and secure. When moving applications to the cloud, consider the following:

1. For application development, choose PaaS; the infrastructure is managed by a vendor giving you more time to focus on your application code.
2. For application development, choose a platform enabled for both manual and automatic horizontal scaling of your application.
3. For green-field applications, design apps that are stateless.
4. For legacy or existing applications, choose a PaaS provider that supports both stateful and stateless applications.
5. Choose a database that is scalable and located on a separate server/container from your application code so that you can scale the database independently.
6. Choose a cloud provider that enables you to both deploy and scale your application infrastructure across multiple geographies throughout the world.
7. Develop using REST-based web services.
8. Choose a cloud provider that meets all of the requirements above with the added feature of integrated CI/CD tools to the platform.
9. Avoid vendor lock-in.

Related topics

- Explore developerWorks Cloud computing, where you can find valuable technical resources and basic cloud computing concepts.
- See "The best of developerWorks cloud computing: The old, the new, and the blue" (developerWorks Cloud team, developerWorks, October 2014): Access some of IBM developerWorks' best cloud-related content.
- "Cloud computing fundamentals" (Grace Walker, developerWorks, 2012): Learn the basics of cloud computing.
- Try IBM BlueMix.