Build Cloud Test Application & Research

Daniel R. Cender

Grand Canyon University: CST-323-O500

Nov 2, 2020

**Test Application**

**GitHub URL:** [**https://github.com/DanielCender/CST-323-Test-App**](https://github.com/DanielCender/CST-323-Test-App)

The test application for this course activity set is still in development. It will be a multi-page lite-blogging application (think of Twitter more than WordPress), connected to a MySQL database. It’s being developed using NodeJS and the React framework, with Bootstrap for a consistent style system.

Users will “claim” a username for the time they are using the app. After they logout, that name will again be available for others to write posts under (or delete old ones). The main feed page has a basic text editor form and a scrolling view of the latest posts on the site. By clicking on the user’s username near the Logout button, they navigate to a view where they can manipulate all their user’s posts.

**ER Diagram**

As of this milestone, the test application requires only one database table to function.

Graphical user interface, application

Description automatically generated

**UI Designs**

Some basic preliminary designs made in Canva.

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text

Description automatically generated

A picture containing graphical user interface, text

Description automatically generated

There will be an “About” page, including a feedback form that will create a comment in a different database table.

Cloud Computing Research

**Weighing the Models**

Below is a compare/contrast of the different application deployment models available.

* **Public Cloud**
  + **Advantages**
    1. **Utility Pricing:**Users only pay for their usage of their application resources. No unused storage space or CPU time is consumed at their expense.
    2. **Elasticity:**The public cloud deployment model affords scaling up to powerful limits which would be unaffordable without the cloud model. The cloud user can allocate more resources or hard limits for spikes in traffic, but only access and pay for them when needed.
    3. **Core Competency:**By outsourcing the management and installation of all infrastructure and hardware to a 3rd party, the user can spend more focus and time on building their solution. From a business standpoint, it requires less staff members and less time-based roadblocks to launching a product/service.
  + **Disadvantages**
    1. **Control:**The end-user is at the mercy of the cloud vendor for the restoration of service in the case of failure. On the same line of thought, the user is limited in what they can deploy by the current offerings of the public cloud sector and individual vendors.
    2. **Regulatory Issues:**Depending on what application is being built or what countries of operation are at play, the developers may be limited in how or in what way they can deploy their service, due to legal regulations. While some cloud auditing solutions exist, many companies are required to use hybrid solutions to avoid these issues.
    3. **Limited Configurations:**The configuration limits are set by the individual public cloud vendor. There may be instances where very heavy computational requirements are needed, but cannot be met, by the current preset configurations of the cloud deployment sector.
* **Private Cloud**
  + **Advantages**
    1. **Less Regulation Issues:**While not all regulations, especially in regards to user privacy, are circumvented, using a private cloud deployment avoids some limitations of current public cloud offerings. If the private cloud is entirely on-premise, then the end-user has all the control they need and can afford.
    2. **Less Hardware Restrictions:**If the end-user is managing an on-premise deployment, they can install any number of servers of any capacity to suit their needs, so long as they have the money to afford it.
    3. **Greater Security Controls:**If the end-user is hosting on either an on-premise or hosted private deployment, they still retain the ability to implement greater varieties of security controls, since their resources are not being shared by any other cloud customers.
* **Disadvantages**
  1. **Reduced Elasticity:**When a solution is deployed on a hosted deployment or on-premise, there is a significant delay between spikes of traffic and the availability of the power to handle the spike. The user would have to physically install more infrastructure or connect with their cloud host to install it for them.
  2. **Higher Overhead:**When the end-user is responsible for purchasing all hardware and installing it themselves, they are responsible for the full cost and lifecycle of that equipment. They do not get the luxury of paying only for their consumed CPU time and disk space on a wider public cloud.
  3. **Slower Development Time:**When the agility and elasticity of your infrastructure suffers, your development will face hardware roadblocks potentially that it wouldn't with using the public cloud, in most cases.
* **Hybrid Cloud**
  + **Advantages**
    1. **Utility Pricing:**With part of a product in the public cloud, the user receives a great added benefit of consumption-based pricing.
    2. **Elasticity:**The beefiest parts of an application can still be built and offloaded to public cloud architectures while preserving the most sensitive and precarious of services for on-premise or hosted cloud offerings.
    3. **Security:**A user can offload high-risk data and operations to on-premise solutions, while still reaping the benefits of the configurable public cloud.
  + **Disadvantages**
    1. **Higher Costs than Pure Public Cloud:**Although this deployment model isn't as costly as the private cloud, it still requires a higher overhead cost and devotion to maintenance than the pure public cloud deployment model.
    2. **Greater Development Overhead:**Increasing the complexity of your cloud deployment model means a development team will require more time to design/build/test/secure the system.
    3. **Higher Barrier To Entry than pure Public Cloud:**Similar to the increased development time, a team will require a wider knowledge base or consulting outlets to piece together their entire solution, depending on their budget and current team size.

**SaaS Platform Comparison**

* **Salesforce**

This platform has transformed the CRM landscape with its SaaS model and extensive use of 3rd party integrations and API offerings. There are many developer courses and certification programs on just learning how to develop using Salesforce's tools.

* + **Advantages**
    1. Their vast ecosystem of connected service and product offerings is unmatched by any on-premise solution on the market right now. It doesn't have to just be a CRM; it could also help a user build an e-commerce platform or develop a simple mobile application with no code at all.
    2. As the most popular CRM in the world, a user benefits from their customer support experience, documentation, case studies, fellow Salesforce user groups/classes, and conferences. Someone would essentially be able to get an answer or solution to a product-based issue in a quick time frame. With an open-source or on-premise solution, sometimes support moves slower if it requires operating remotely on your installation device.
    3. Ease of maintenance. This entirely cloud-based CRM solution requires zero maintenance or setup where infrastructure is concerned. With on-premise, an entire team (or at least an individual) would be needed with high expertise in this area, which may require a range of IT and scripting/development skills.
  + **Disadvantages**
    1. Data does not reside only/primarily on the user's owned servers or devices. Salesforce ultimately manages its customers' data, so it may not be eligible for a full local backup-and-restore method of disaster recovery. If their services suffer outages, then many, or all, of their customers could be left without a way to collect revenue, data, or new leads.
    2. Because Salesforce attempts to sell their suite of tools, a customer may end up buying more than they intend to consume. Depending on the size of the company or the operation, digging into this platform may be overkill for their needs or budget.
    3. Minimal customization for a company's needs. If a company is on the larger end of the spectrum, like Apple or Amazon, there may not be a platform that can handle their needs entirely to their liking. They might prefer having absolute control over an internally-based CRM and data-collection tool set, which they can extend with their own developers at any time. I could only find Corteza (<https://cortezaproject.org/>) as an example of an open-source or extensible CRM. Something like this may be preferable if customization is key for a company.
* **Slack**

This cloud-based messaging and collaboration service has been transforming and reshaping how remote communication has been conducted with distanced teams. They've even spawned a new niche market of apps that position themselves as a more lean or cheaper "Slack alternative."

* + **Advantages**
    1. Slack offers robust, easy integrations and APIs which allow retrieving and manipulating data however a company might wish. There aren't many alternatives, especially not on-premise ones, which offer this same level of ease for 3rd party integration.
    2. The service is universally available on all devices or through the web browser, so it "goes with you" everywhere you need.
    3. Just like with the Salesforce example, there's no need to manage messaging servers or infrastructure. Just sign up, invite your team, pick you plan, and it just works.
* **Disadvantages**
  + 1. Lack of direct control over data integrity and retainage. An alternative on-premise messaging service like Mattermost (<https://mattermost.com/>) allows firms to always control their communications data directly.
    2. The cost of an enterprise-level subscription for Slack or a related SaaS is expensive, even for some mid-level company budgets. Many of them already use email or a more complete suite like Microsoft Office365. It'd be a waste to spend a ton more on just one app, especially since Microsoft launched its competitor: Teams.
    3. Higher risk. Slack is still not largely profitable as a company. It's experienced a tenuous life cycle as a true startup, and so it hasn't always had the most stable financial situation. Technically, it still doesn't. Hosting your own on-premise communications solution would tie that utility's stability to your own company, not a startup you have zero control in. Investing in startup technologies and tools can sometimes backfire and cause more problems later if that company goes under.

**IaaS Vs. PaaS Deployments**

Two common methods of deploying cloud applications involves either Platform as a Service or Infrastructure as a Service. IaaS allows developers to allocate themselves full use of a virtual server space, just as complete as a physical server they might install themselves and manage. They are still required to manage the operating systems, middleware, networking, storage devices, and servers. PaaS, however, goes a level higher than IaaS. With PaaS, you can skip managing the networking, devices, etc. PaaS allows developers to just build their application and deploy it to a service which will handle device management and autoscaling for them. One clear benefit of PaaS over IaaS for my test application is its ease for developers. There’s far less to configure and maintain, even though it’s the less mature cloud technology.

Thankfully, a small application like this would not need the kind of scaling that would render PaaS useless. Large services, like Facebook or Twitter, require IaaS, because their needs are too great for a simplified platform to handle at scale. There’s also a higher initial cost to provisioning IaaS resources than PaaS resources. Platforms like AWS AppSync or Azure Functions are highly consumption-based in their pricing model and affordable even at high volume. So, another impactful difference here would be a decrease in cost if I choose PaaS over IaaS.

References

Kavis, M. (2014). *Architecting the cloud: Design decisions for cloud computing service models (SaaS, PaaS, and IaaS)*. Hoboken, NJ: Wiley.