Future Cloud Computing and Cloud Research

Daniel R. Cender

Grand Canyon University: CST-323-O500

Dec 8, 2020

**Cloud IDE Research**

**Features**

GitHub CodeSpaces is a new service offering from GitHub. While in its testing phase, there is no charge for use on personal repositories. It runs a copy of VSCode on a Docker container service, with some additives like port forwarding to fill out its feature set as a fully usable IDE. VSCode extensions can be installed just like on the desktop version. These CodeSpaces are repo-specific and shutdown after 30 minutes of inactivity.

**Workflow Use Cases**

Below is a breakdown of how GitHub CodeSpaces allows me to perform my necessary development tasks on my desktop IDE for the course project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Supported (Y/N)** | **Observations on Usability and Productivity** | **Other Notes** |
| Opening local folders to grab snippets of code | N | The editor is scoped to just the repository contents. |  |
| Editing editor settings | Y |  |  |
| Installing external extensions | Y | Same extensions are allowed as in desktop VSCode. |  |
| Saving code to version control | Y | Using Git cli tools or the GUI editor tab are options. |  |
| Running development processes/servers | Y | CodeSpaces handles port forwarding automatically for easy testing. |  |
| Use keyboard shortcuts | Y | Supports many of the exact same shortcuts as VSCode. |  |
| Install and manage app dependencies | Y | Full use of a bash terminal on 40GB of space. |  |
| Connect to Local Database instances | N | Port forwarding seems limited for this purpose. | For testing, would need externally hosted database. |
| Manage different versions of lanaguages/tools/CLIs | Y | In the case of the test app, CodeSpaces supports NVM (Node Version Manager) out of the box |  |
| Run Builds and Initiate Deployments | Y | Through use of VSCode extensions and the terminal, these are both possible. |  |

**Cloud IDE Analysis**

Where once desktop IDE’s where the only avenue for developers to begin and develop their projects, new iterations on that concept have risen that open new options. Through GitHub CodeSpaces, Theia, Codeanywhere, or many other options, developers can now choose to develop apps entirely from their web browser. These options are limited in computing power and tooling support, of course. Yet, it opens up possible time savings for development teams and cost savings for management.

For this course’s test application, some fairly standard tooling is in use. At a cursory glance, there does not seem to be any tooling or process hindrances that would come up from transitioning entirely to a cloud-based IDE like GitHub CodeSpaces. The one exception is some issues with port mapping that arise when a development build of the test application attempts to connect with a MySQL database on local port 3306. The port forwarding doesn’t reverse-map that to port 3306 on my personal machine. There may be ways developers could install a MySQL instance in the CodeSpace and run a database on the same machine as the development build, alleviating this issue. I did not have the time to further explore this option, however.

It’s clear that fewer on-premise teams could replace their local development systems with these cloud platforms, given their need for ways to interact with the bare machines themselves. For developers more entirely focused on cloud services, however, there is hefty potential for moving completely to a cloud IDE. All of the features and capabilities that I noted as a clear requirement for this course’s application are available through GitHub CodeSpaces: augmenting editor settings & extensions, running development terminal processes, manage and install app dependencies, connecting to port-bound services, managing different CLI apps, and running builds or deployments.

Every team or department will need to carefully evaluate their development processes in order to determine the ultimate value of transitioning to a cloud IDE. Some local testing setups may be unfeasible without some extra setup or installing alternative systems to prop up existing processes. A lot of flexibility exists with the use of GitHub repository services and GitHub Actions which may be able to fill out a team’s development pipeline to include full Continuous Integration/Continuous Deployment attributes. For the test applications in this class, GitHub CodeSpaces would be more than sufficient to replace the current desktop Visual Studio Code environment, given that the database was already deployed and accessible from a 3rd party PaaS or IaaS service.

**Cloud Computing Research**

**DevOps Questions**

1. What is a feature flag or feature toggle?

A feature flag is a method used by developers and DevOps administrators to set which features should or should not be included in an application build or deployment. Using feature flags allows developers to push out new features to a specific group of users or none at all, should the need arise.

1. What is A/B testing?

A/B testing is a process of sorting out the best design option for a new feature in an application by presenting (traditionally) two options to end users or testers for evaluation. The option which receives the most positive reception is then pushed forward or iterated upon.

1. What is continuous delivery?

Continuous Delivery is the process by which developers can automate their deployment pipeline to transform their source code into a deployable application or service set.

1. What is continuous integration?

Continuous Integration is the process by which developers can push changes to their version control system on a project and see how those changes interact with testing suites, quality assurance automation, and build pipelines.

**Cloud IDEs**

Cloud-based IDEs have become more popular lately as Internet latency has decreased. The other day, I was looking over a new cloud IDE from GitHub called CodeSpaces (currently in a beta-testing phase). Given the cost of the lowest-tier workspace, which is roughly $0.085 an hour to run, it would cost a company far less to set up a developer in one of those environments than on an expensive new machine, even if they worked for the company for a few years.

From a business perspective, this is certainly easier to cost for the sake of budgeting than purchasing machines and accounting for damage and depreciation over time. It would also be easier to get new developers set up with team-standard tools if there is a cloud-configured environment that can be created and reset whenever needed.

* **GitHub CodeSpaces**: This is a new service from GitHub which will be launching soon. A strong benefit of this tool will be its tight integration to source control with GitHub, and also CI/CD pipelines through GitHub Actions. This is really just another step towards allowing developers to run their entire development cycle through just GitHub tools and services. Their CodeSpaces can be configured by the contents of the repo being worked on, giving a greater sense of consistency to a team's working environment.
* **Theia:** Theia is a browser-and-desktop compatible, extendable code editor that was built to be vendor-neutral and open-source. It contains many of the design features of VSCode, going so far as to support VSCode extensions. However, it's far more customizable for a company's individual needs. This editor would be a strong option for larger development firms where they have the resources to define their own editor needs and set up a standard cloud or desktop executable. The costs associated with Theia would be heavily dependent on a company's needs and how they host the system.

A clear downside to these IDE options is that they require internet access to work constantly. If the teams could ensure they have strong and reliable connections to the Internet, then there's little reason they couldn't replace their local development environments with one of these options (given their projects suit the medium). More intensive projects like game development likely won't move to cloud-based IDE's anytime soon.

**Scrum Retrospective**

There are a few points to note about the design of this class.

**The CLC**

Honestly, I don't think this class should have a CLC project at all. There should be forum spaces where students can share resources or insights related to the assignments. Yet, I think placing a CLC as the bulk of this class's point accumulation is almost cruel.

Granted, team projects are the vast majority of "real life" experiences we'll have in development in the workplace. But we'll hopefully be working with people paid to help us. There's simply no guarantee that a team will be involved or experienced enough to get the later milestones finished completely.

Personally, my personal activity assignments suffered majorly as a result of having to prioritize the CLC milestones. I think I would have been more successful in actually learning some CI/CD concepts or standard tools *if I was in more complete control over my ultimate success or failure in this class.*With how it's structured currently, I feel like lots of students probably come through this course and are disadvantaged disproportionally by the roster at times.

**Materials**

The deployment guides were nice to have, I think they are fairly accurate to how the Cloud services work to this day. With how my team organized our CLC project, none of the deployment guides really matched to a T, making the process more difficult. I think that's partly due to the difficulty of organizing exactly how an app should deploy its service API and DB connections, especially on limited time each week.

**Course Substance**

For a class focused around deploying applications to modern cloud solutions, there were little rubric-defined areas where we were truly graded on whether we were ever able to successfully configure different types of cloud deployments or piplelines. Continuous Integration and Continuous Deployment are incredibly important concepts, and yet they received very little attention or build throughout the class time. Focusing on learning a couple notable tooling examples for these best practices would have helped in developing the test applications.

With the focus seemingly shifted from DevOps and cloud infrastructure to building test applications, more time was spent fighting with UI code and frameworks than was with actual cloud or CI/CD tooling.

References

American Psychological Association. (2010). *Publication manual of the American Psychological Association* (6th ed.). Washington, DC: Author.

Daresh, J. C. (2004). *Beginning the assistant principalship: A practical guide for new school administrators*. Thousand Oaks, CA: Corwin.

Herbst-Damm, K. L., & Kulik, J. A. (2005). Volunteer support, marital status, and the survival times of terminally ill patients. *Health Psychology*, *24*, 225-229. doi:10.1037/0278-6133.24.2.225

U.S. Department of Health and Human Services, National Institutes of Health, National Heart, Lung, and Blood Institute. (2003). *Managing asthma: A guide for schools* (NIH Publication No. 02-2650). Retrieved from http://www.nhlbi.nih.gov/  
health/prof/asthma/asth\_sch.pdf