HW 02 Docker and Configuration Management

Tarik Kdiry
SSW-567
September 8, 2018
I pledge my honor that I have abided by the Stevens Honor System

1) Integration

Often times in integration, developers on the team have a large variety of tools they use to write their individual contributions to the master branch. This can be an issue when integrating each developers contribution if their tools and dependencies do not interact well with other developers' tools and dependencies (rails gems and flask modules are notorious for this). Docker ensures the deployment at hand is ready to integrate by containerizing services and allowing developers to do everything such as debugging and testing under one environment.

2) Testing

A common issue in testing occurs in situations where a project is too big and far-gone in its development cycle that testing small or older components may be difficult due to things like coupling. Docker provides a solution by compartmentalizing app components into containers that allow for easier troubleshooting and testing at a smaller scale, regardless of the size of the project. It will also allow developers to replicate the development cycle of that app component in order to better identify weak points.

3) Deployment

A common issue that occurs during deployment involve integration. With integration, comes inconsistent environments and dependencies that may cause trouble during deployment and scaling. Docker ensures that all containers within the project have consistent configurations and dependencies from start to finish. Along with this, necessary changes can be made within Docker at any time, and tests can be made subsequently for multiple servers under the same environment.

4) Customer Support

Sometimes customers can add new changes or file new complaints at any point in the development cycle after deployment. The biggest issue is curing their problem without affecting other app components (due to coupling), along with adding new changes to a certain app component includes having to go through the whole project in order to do so. Since app components are containerized within Docker, a customer change can be made without having to go through every app component and fixes can be remedied in a similar fashion.

5) Internal and External Hackers

Hackers by nature depend on the vulnerabilities of a program in order to do their damage. Whether that is to hijack data, corrupt a program, etc. Vulnerabilities are very difficult to discover even after the damage is done, let alone before it. Since components are containerized within Docker, debugging becomes a lot easier since each component can be scrutinized individually, therefore shining a brighter light on vulnerabilities. Version control tools such as Git, along with Docker, have systems set in place in order to notify you of vulnerabilities and their locations before an attack occurs. This is regularly seen on GitHub.