# Continuous Integration

Continuous integration is a key part iterative development, since it allows code to be deployed efficiently and quickly while providing a mechanism for defects to be caught early in the development lifecycle. To achieve continuous integration, the team set up a Jenkins Server in our AWS environment running on CentOS 7 (http://jenkins.sditdev.local:8080). Jenkins is an open-source continuous integration application that has been adopted widely in the software development community and is surrounded by an active community of users and developers.

Three projects were configured in Jenkins, each corresponding to a BitBucket repository containing the source code for one of the three pools for the challenge. A Docker plug-in for Jenkins was installed in order to allow Jenkins to utilize a Docker container set up with a .Net eco-system including MSBuild. Jenkins was configured to build each project, execute the unit tests, and deploy the compiled projects to their corresponding production servers. Each project was configured to run every hour using Jenkins built-in scheduling feature. The Jenkins setup is depicted in figure x.x:

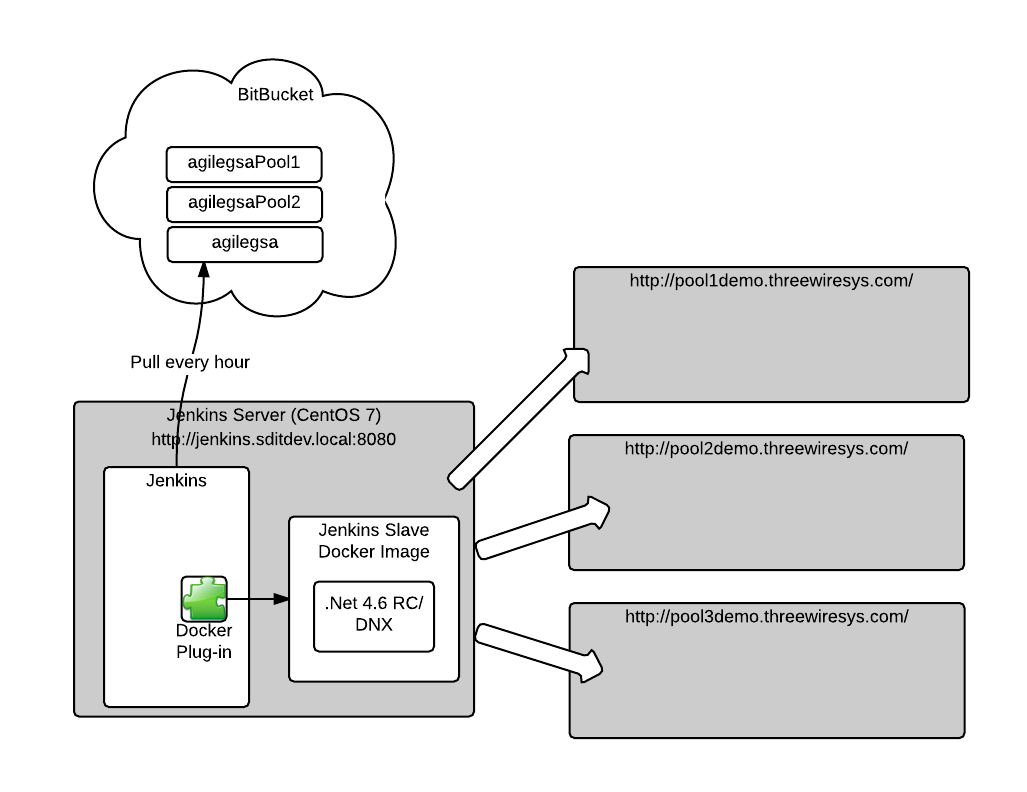


Figure x.x: Continuous Integration utilizing Jenkins

While the team has experience with CI in a .Net ecosystem, the requirement to use open source technologies exclusively for the continuous integration package and server OS presented an enormous challenge for an “MVP” project such as this. These challenges were made even more acute by the fairly recent introduction of Visual Studio 2015 and the DNX runtime environment, neither of which are well-documented or supported by Jenkins community.