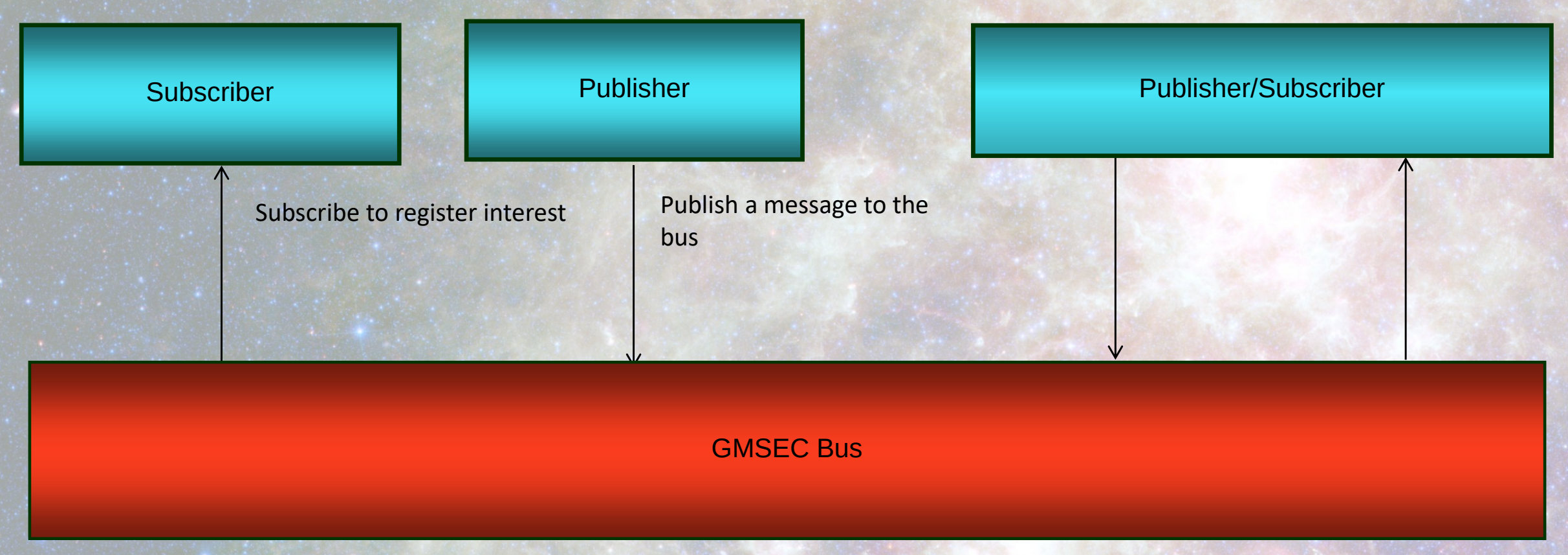
Continuous Delivery Pipeline for GMSEC

Nikhil Mittu

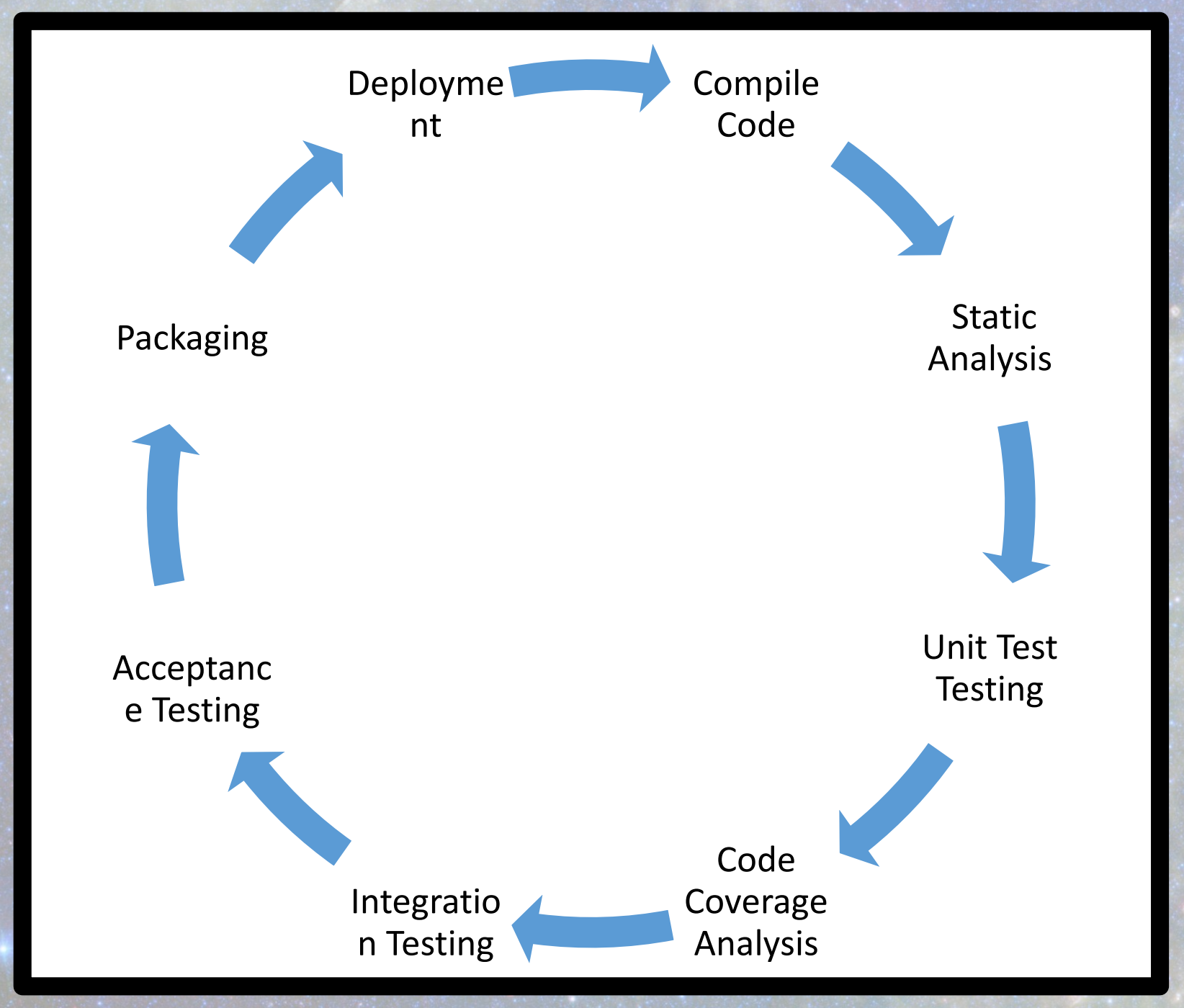
Sherwood High School 300 Olney Sandy Spring Rd, Sandy Spring, MD 20860

Background

The Goddard Mission Services Evolution Center (GMSEC) is a program whose efforts are to coordinate and reduce the cost of the development of flight data systems. The GMSEC architecture allows scalable and extensible ground and flight systems to be built for missions. GMSEC allows for the use of a number of components that can be selected based off the needs of the mission. GMSEC is also extremely flexible in that it allows for the addition, deletion, and exchange of components being used.

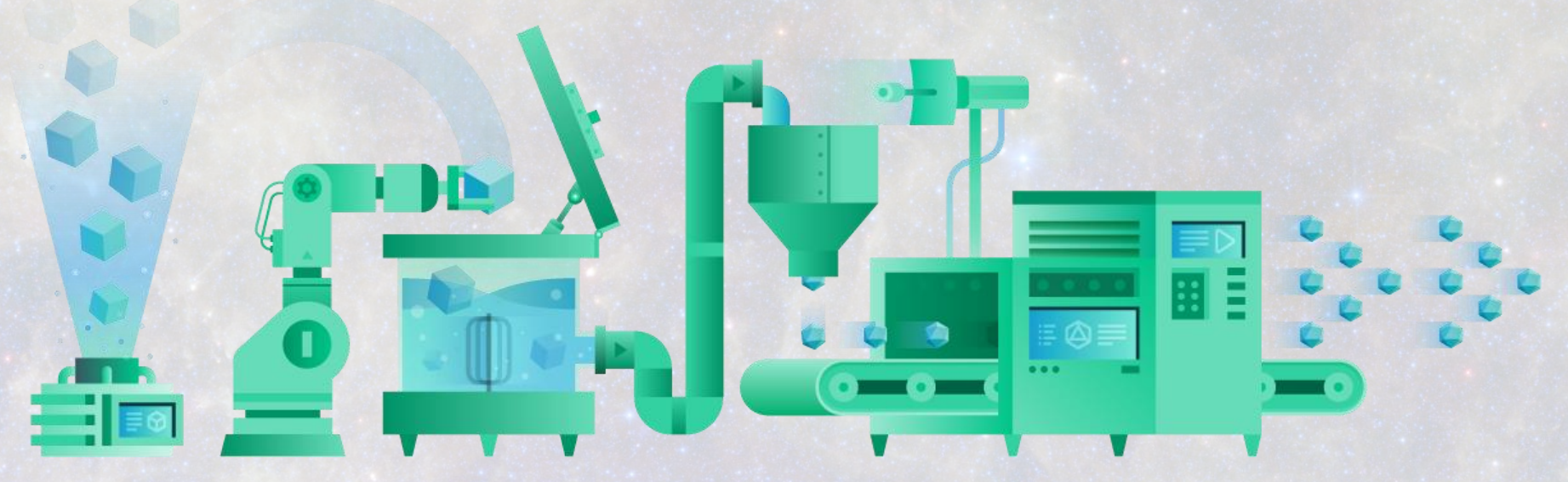


A continuous delivery pipeline provides a streamlined build process that allows a project to be tested and deployed more often throughout the development life cycle. This can help developers find issues and errors in their code quicker because they are able to run a build, to catch issues, whenever they make changes to the code.



Gradle

- The GMSEC API and the GMSEC component are built using the Gradle build system.
- Most of the build tasks that the Components use are common across all of the GMSEC components so a custom Gradle plugin was created that contains all of these Gradle tasks.
- This Gradle plugin is then added as a buildscript plugin to the build.gradle files for each component.
- This means that the code for the tasks is just in one location and changes only have to be made in one location.

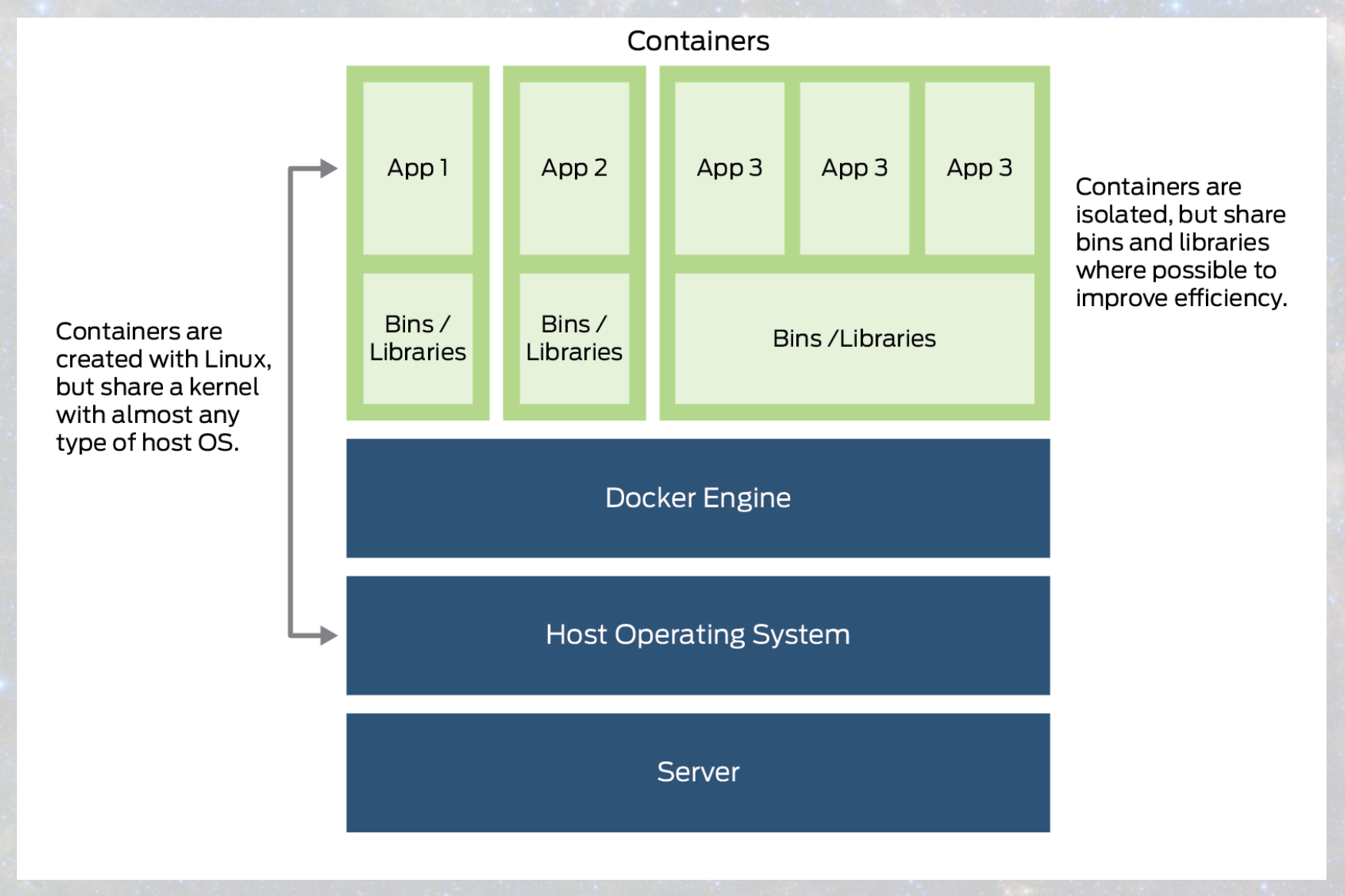


Learning Curve

Built Tool	OS Support	Language Support	Environment Discovery	Dependency Management
Autotools	NO	NO (java is painful)	YES	Only Checks, does not install.
CMake	YES	NO (java is painful)	YES	Only Checks, does not install.
Ant	YES	NO (c++ is painful)	NO	NO (Needs IVY)
Maven	YES	YES	NO	YES (for Java)
Gradle	YES	YES	YES	YES (for Java)

Docker

- Docker containers are used to build and test the GMSEC API and components in a consistent environment.
- This means that the build process will always work no mater how the environment on the host build server changes.
- It will be easier to make changes to the build environment in the future, for example if additional software needs to be installed, because changes only need to be made in the Docker container not on the host.


Containers
App 1
App 2
App 3
App 3
App 3
Bins / Libraries
Bins / Libraries
Bins / Libraries
Docker Engine
Host Operating System
Server
Containers are created with Linux, but share a kernel with almost any type of host OS.
Containers are isolated, but share bins and libraries where possible to improve efficiency.

Artifactory

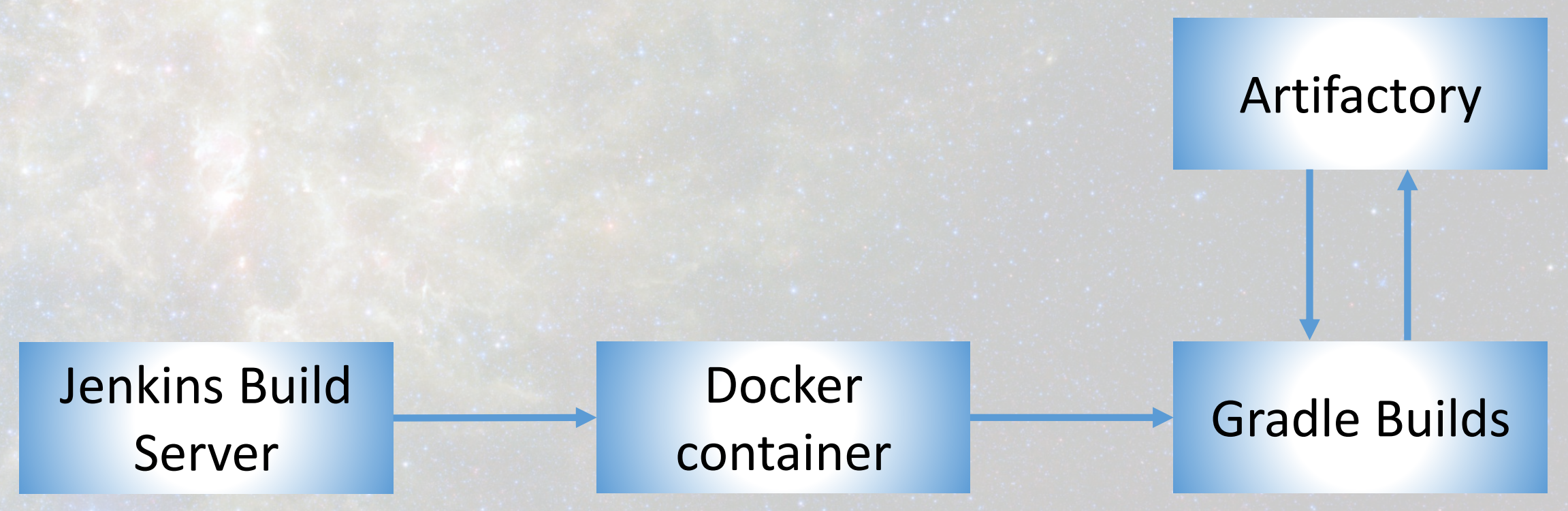
- Artifactory is used as a repository for the dependencies of the API and components.
- Gradle pulls the dependencies from Artifactory at build time.
- The custom Gradle build plugin also has tasks to publish jar, rpm and deb files, produced from building the components and API, to Artifactory.

Jenkins

- Jenkins is used to preform the builds.
- When a build is triggered, the Jenkins build server pulls the latest version of the necessary Docker image from the Docker registry.
- The Jenkins build server then runs the Docker container which builds the GMSEC API or a GMSEC component.

Conclusion

With the new continuous development pipeline the build process for the GMSEC API and components is much simpler than before. Much of the build process is now automated. With the new build process all one needs to do to start a build is trigger a build in Jenkins. Jenkins will then run a Docker container that will run the necessary Gradle commands necessary to build the project. After the build has completed, Gradle then publishes the completed build to Artifactory.



What I Learned

I learned how to use many valuable tools while working on this project such as:

- How to use Gradle and how to create Gradle plugins.
- How to use the Groovy programming language.
- How to use and create Docker containers and images..

Acknowledgments

I would like to thank my mentors Theresa Beech and Rhea Mortam for their guidance and help