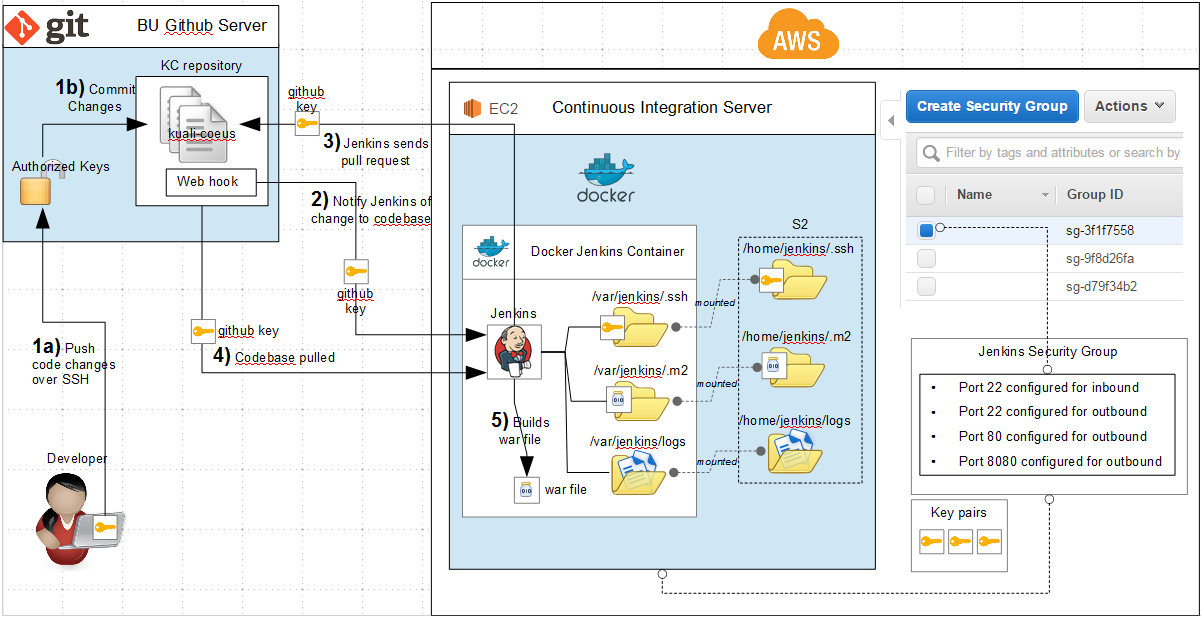
A single EC2 instance acts as the BU Kuali Coeus build server. This server has Docker installed.

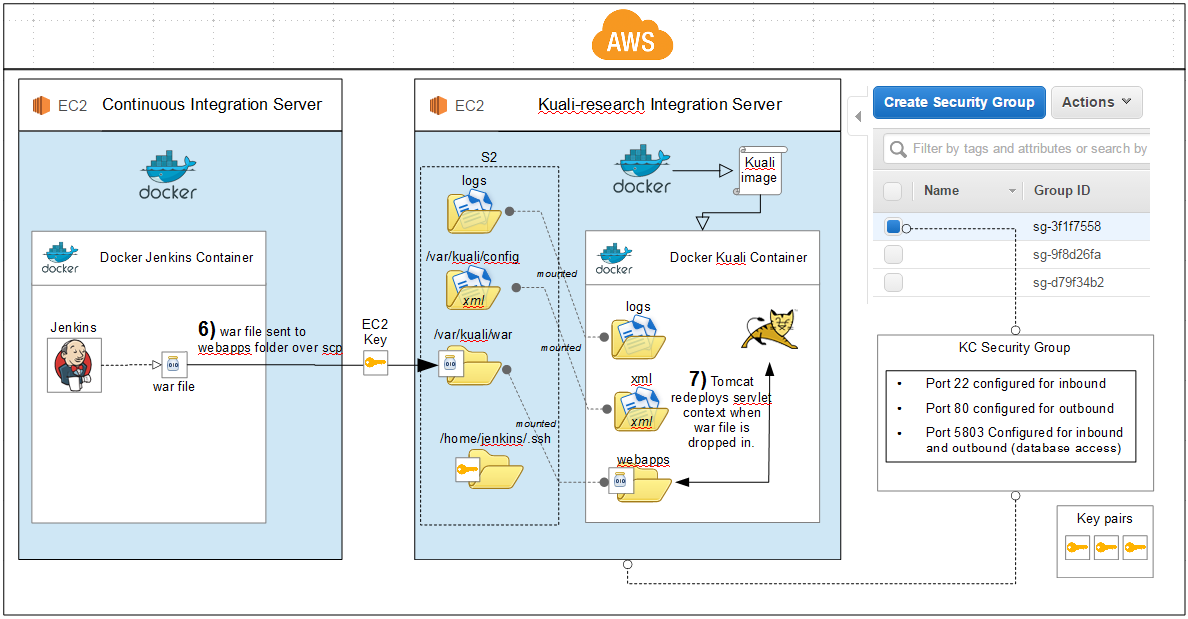
Docker runs a container that virtualizes a Jenkins web site and its required software infrastructure.

Build cycle:

1. A developer pushes code changes to our BU github repository
2. The github repository is configured with a web hook whose purpose is to notify Jenkins that a change to the codebase has occurred.  
   The port over which the webhook is configured to talk to the EC2 instance is a custom port (ie: 8080).  
   The EC2 security group for Jenkins defines this port as accepting inbound traffic only from the BU github domain/ip   
   *(Alternatively, Jenkins can be configured to poll the github repository for changes, making it unnecessary to configure github for access to the EC2 instance)*
3. Jenkins sends a request to pull the new codebase down to a local workspace.  
   This requires that the Jenkins user has access to a private key expected by the github account.  
   This key is in a directory that is mount-binded to a directory internal to the Jenkins docker container.
4. Github authenticates the provided key, and sends the requested data
5. Jenkins builds the new Kuali-research war file.



1. The war file is sent via scp to a directory on the application EC2 instance.  
   To do this:
   1. The Jenkins EC2 instance must have access to a private key that will gain it access to the application EC2 instance.
   2. This key is in a directory that is mount-binded to a directory internal to the Jenkins docker container.
   3. The port range configured for the KC Security group must include the domain/ip of the Jenkins EC2 instance.
2. Tomcat is configured to redeploy any web applications upon receipt of a new war file.



Docker containers can be run so that ports referenced internally are linked to ports external to the container

