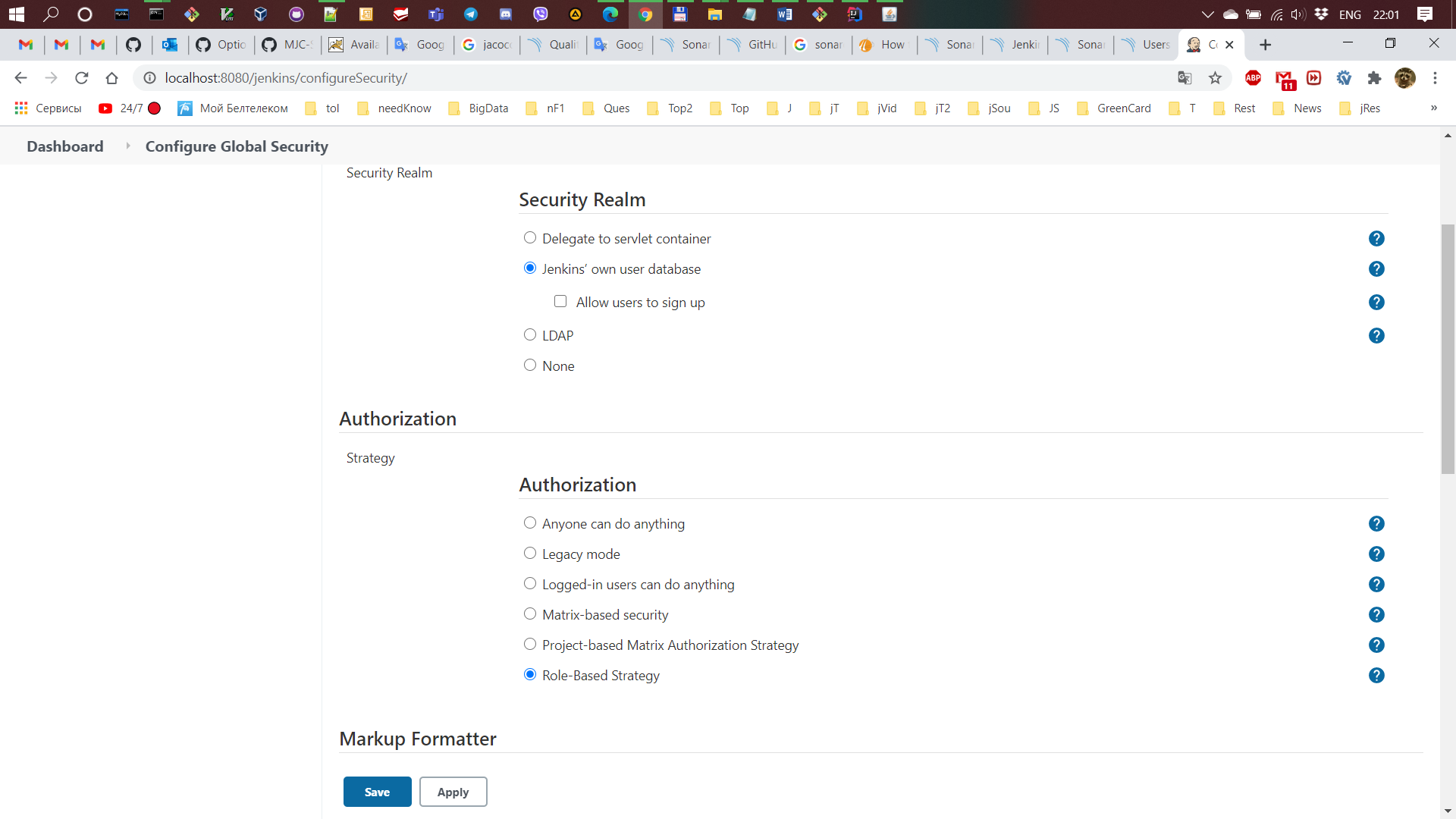
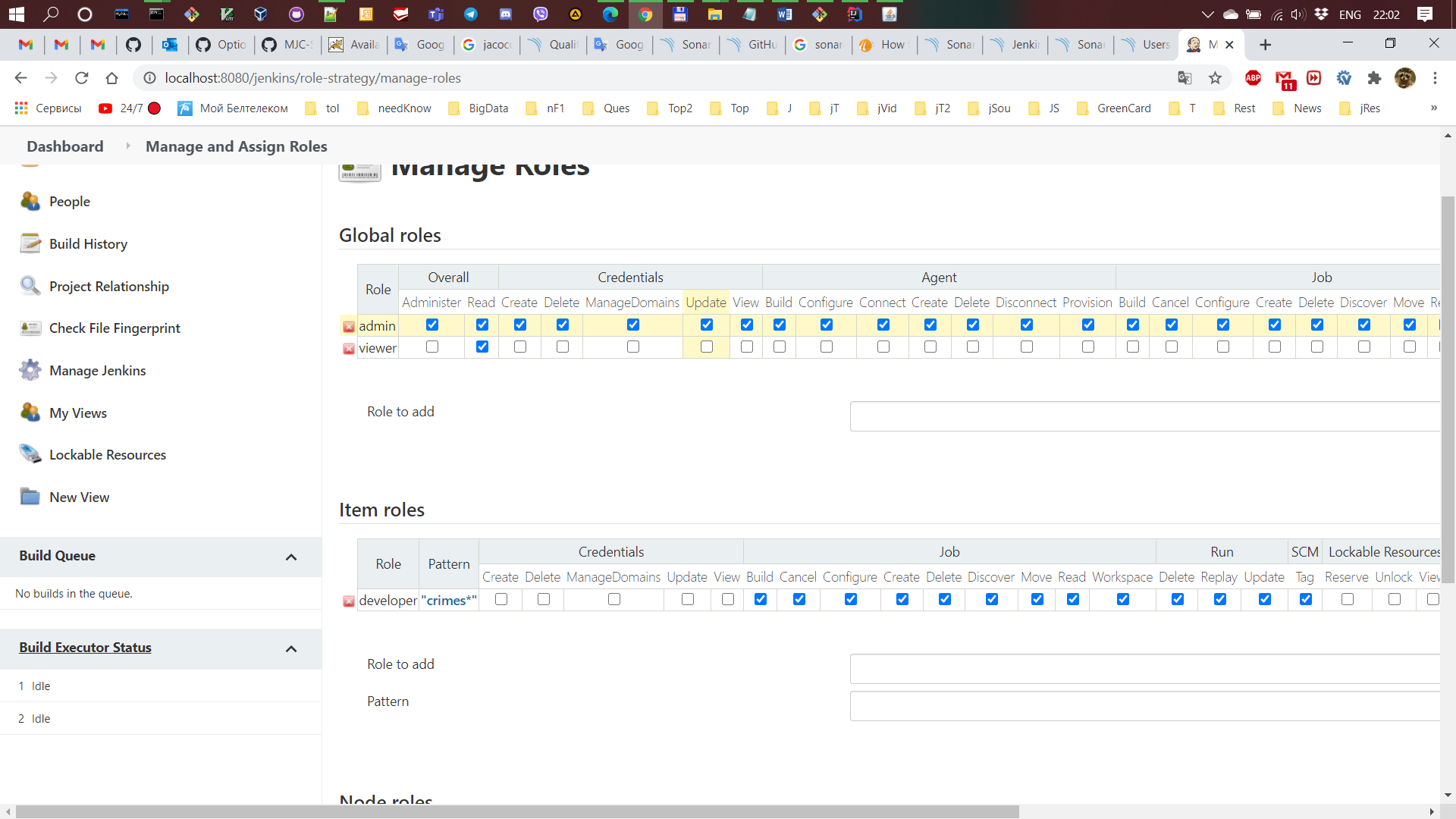
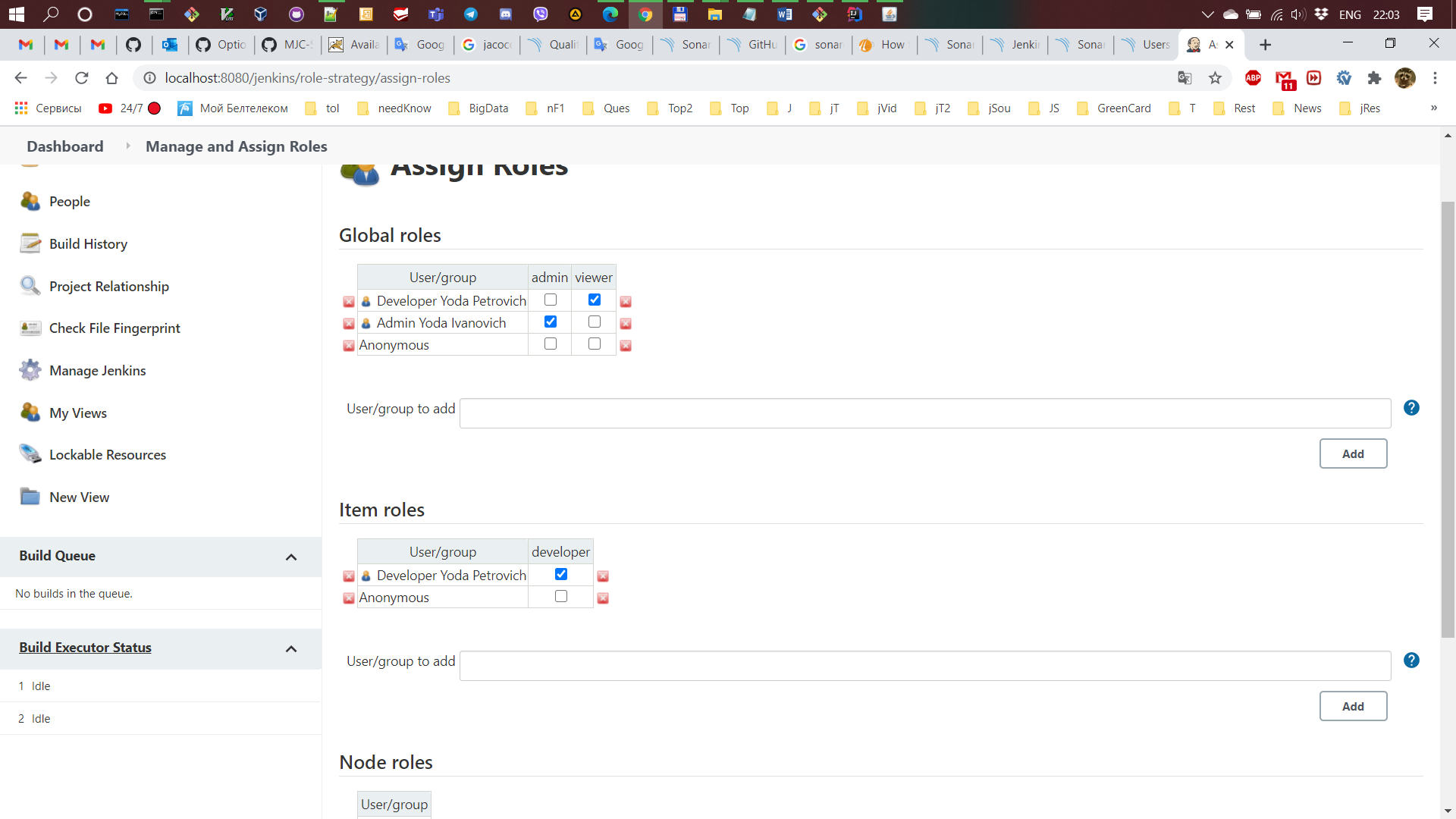
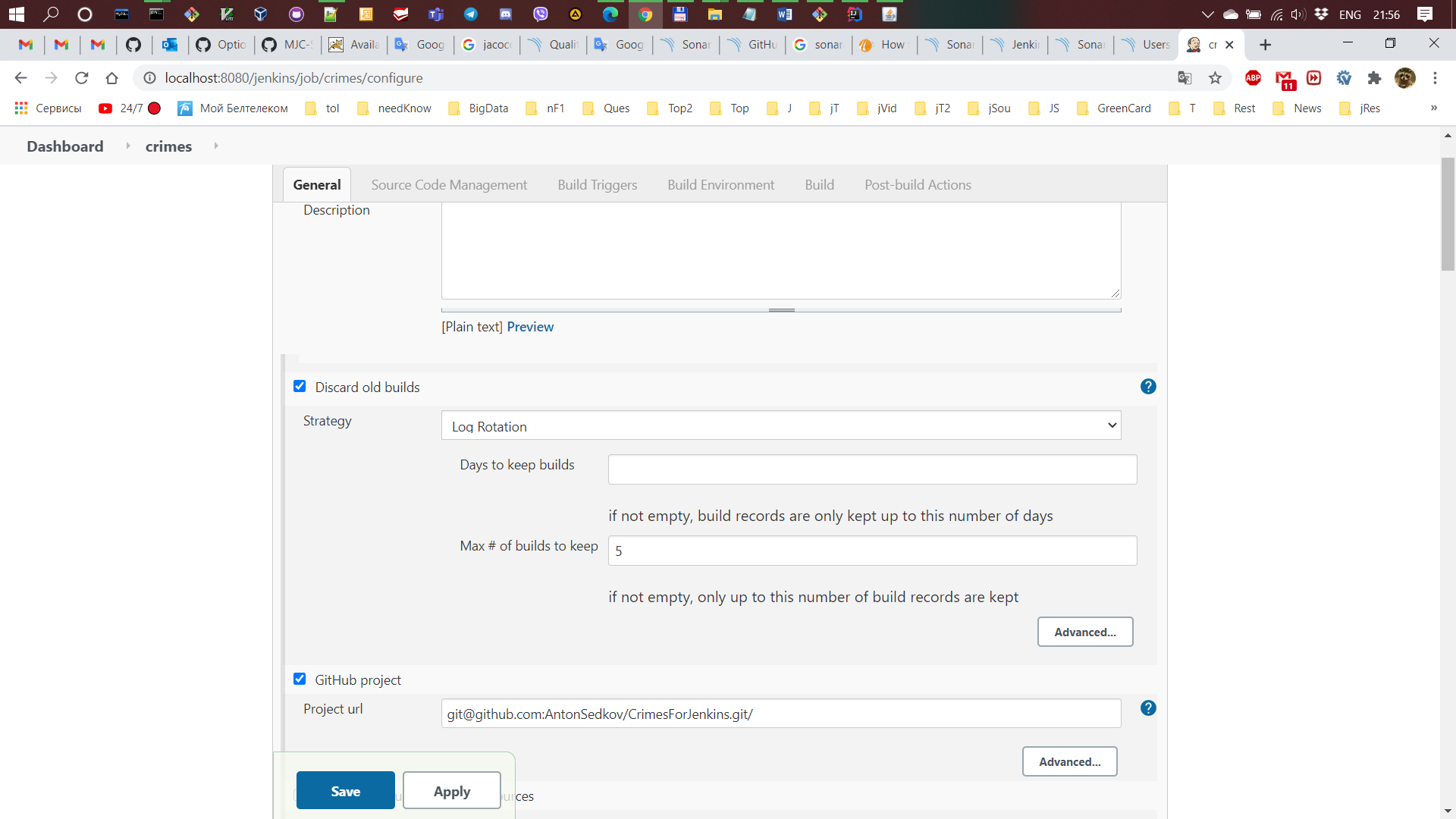
1. Configure Jenkins security (install Role strategy plugin). Remove anonymous access. Create administrator user (all permissions) and developer user (build job, cancel builds). Add Jenkins credentials to Readme file in your git repository.

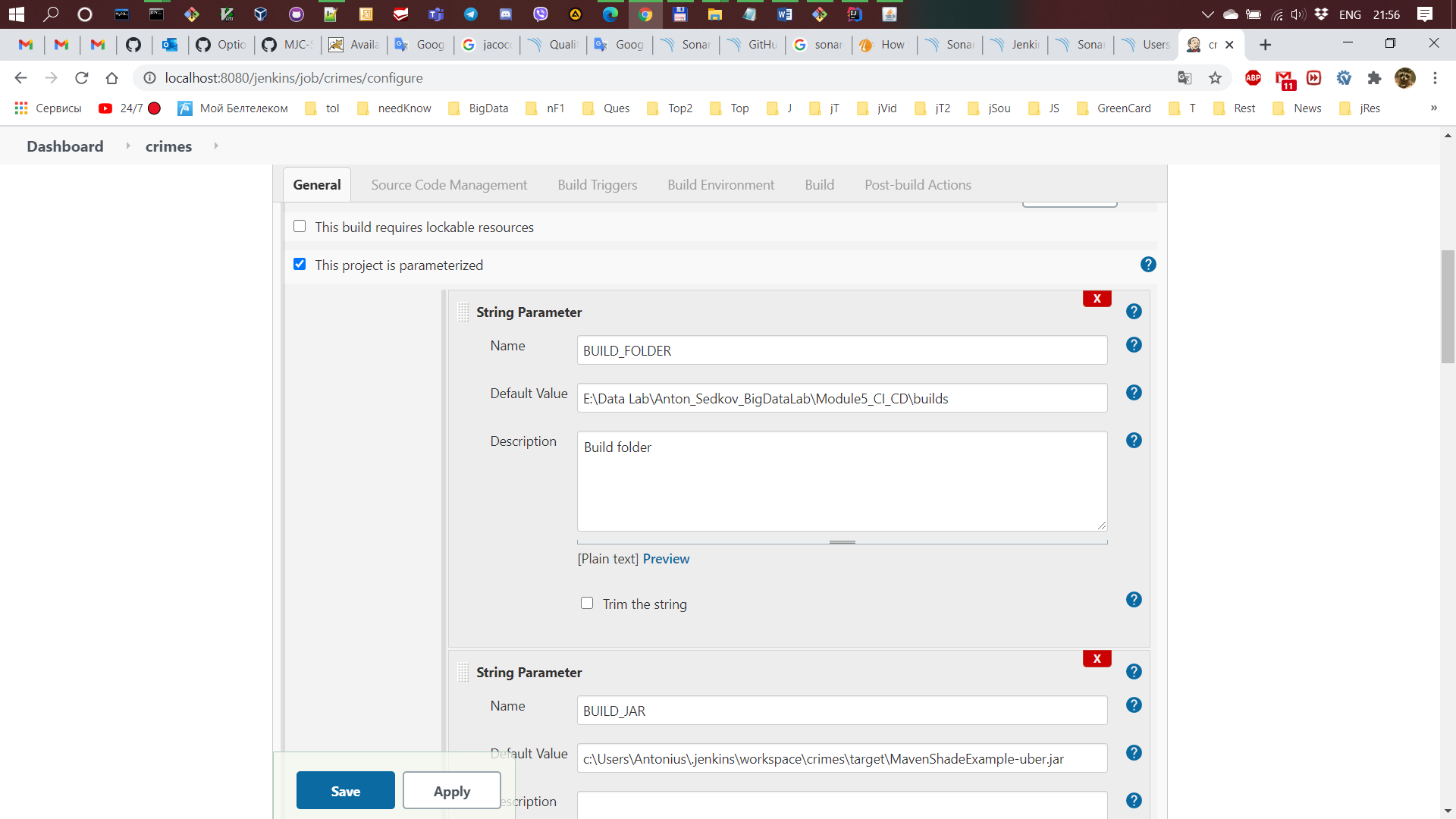


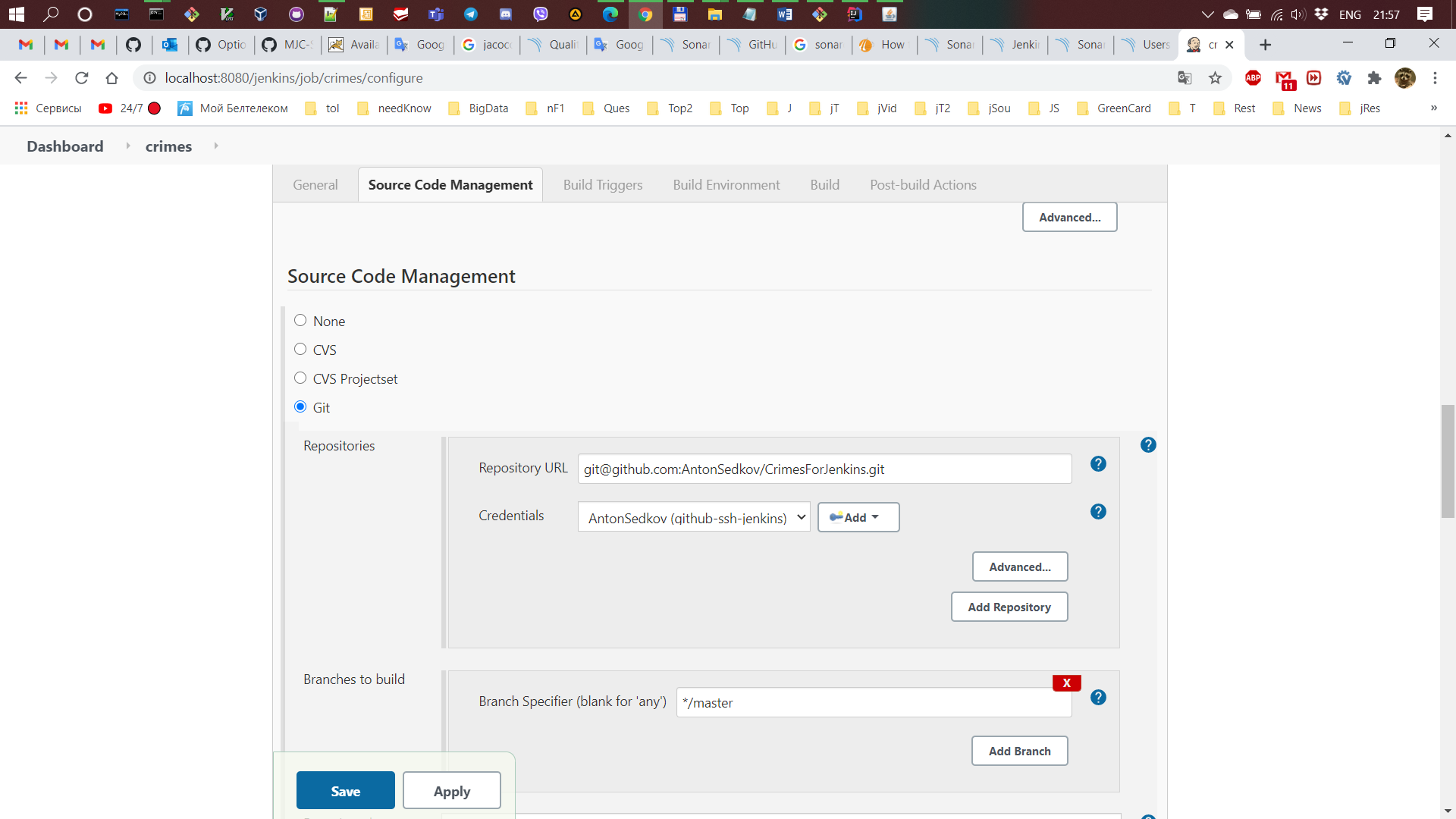


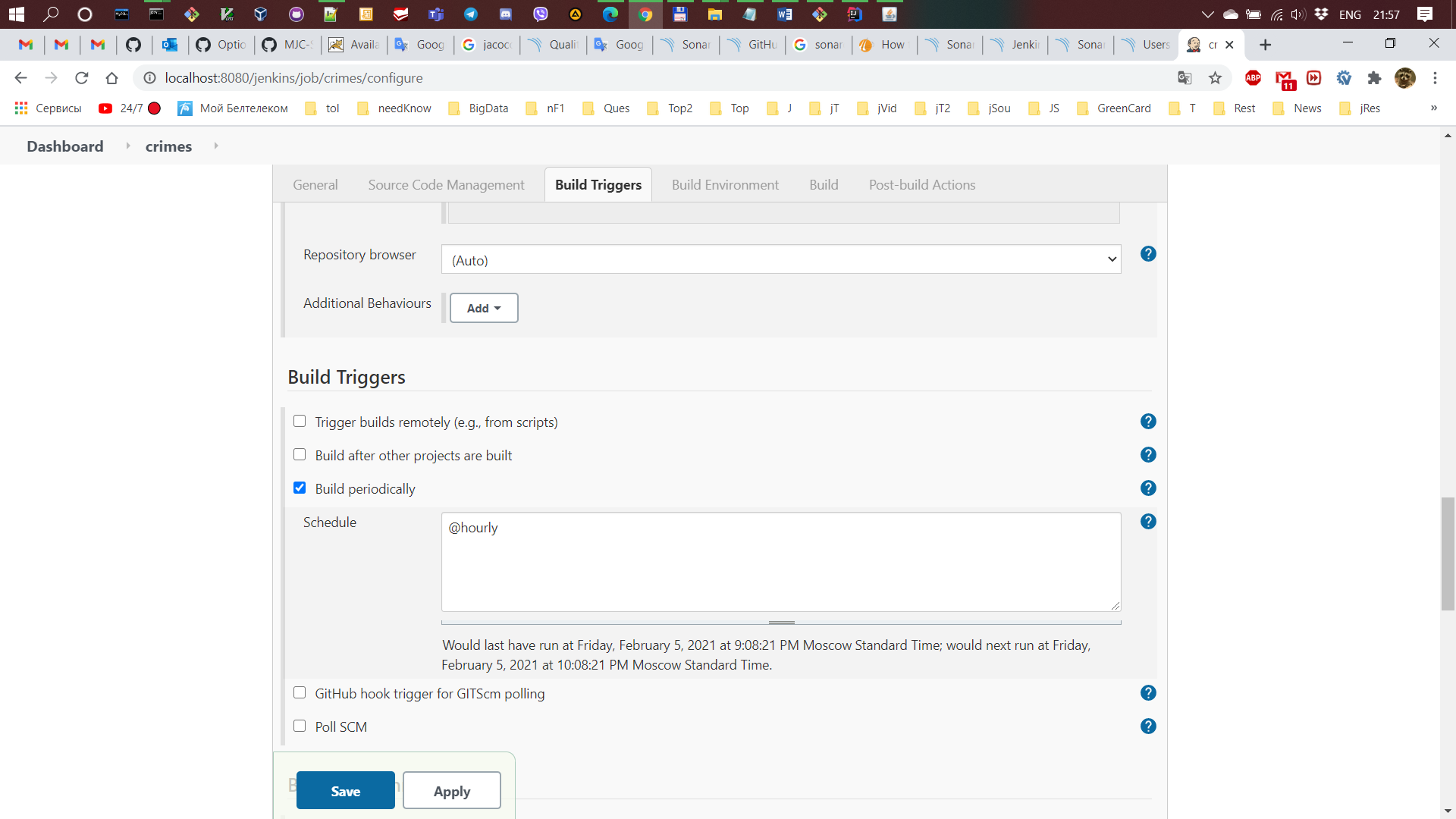


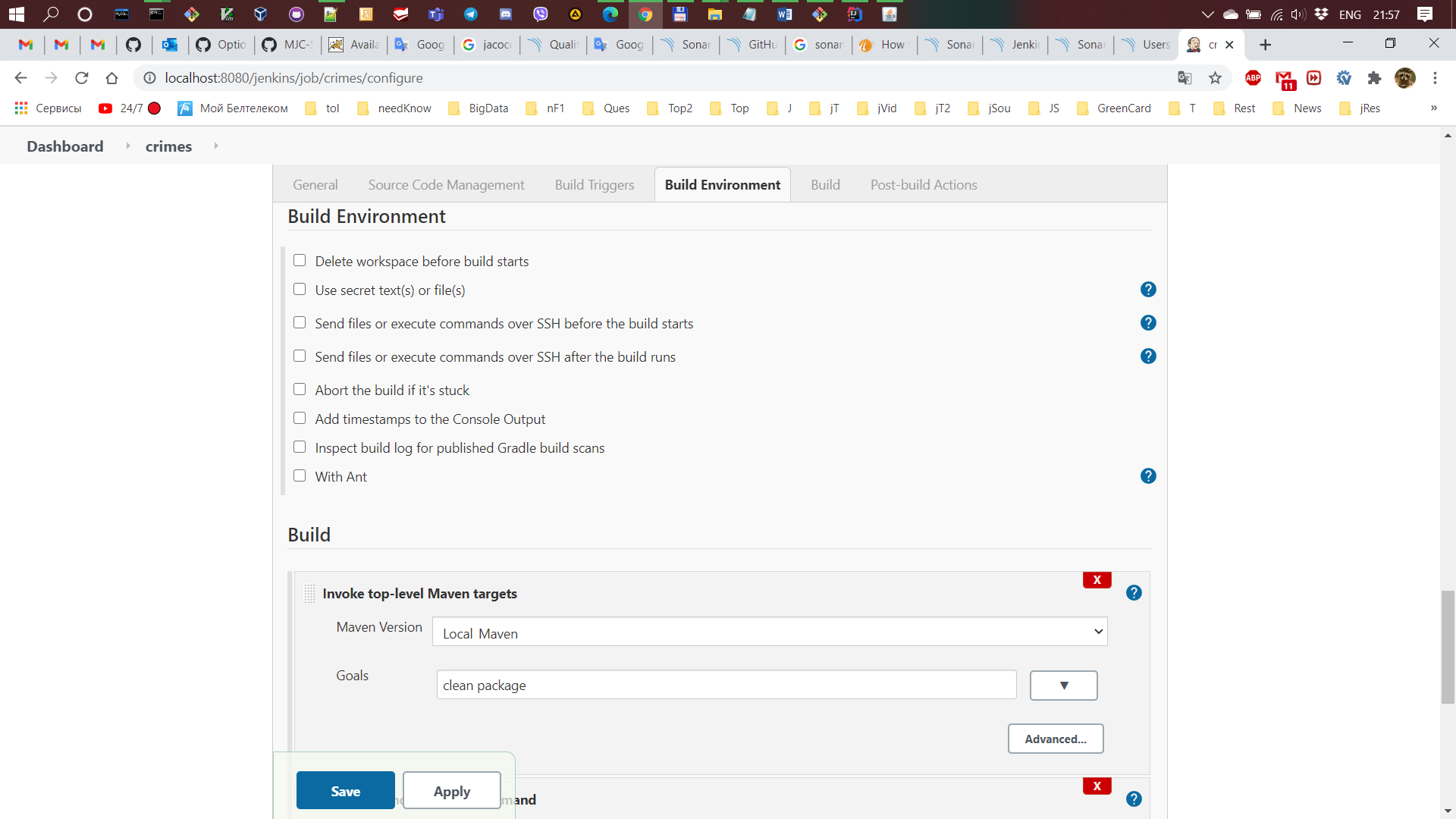
1. Create Jenkins build job. Configure it to checkout your repository, build your application using maven. Save artifact to subfolder of builds' location folder. The parent folder location is a constant configured inside the job. Name of the subfolder defines build version, it should be generated incrementally (example: 1, 2, ..., n) or taken from the Jenkins build number.  
   *Sample structure: ~/builds/1/\*.jar*

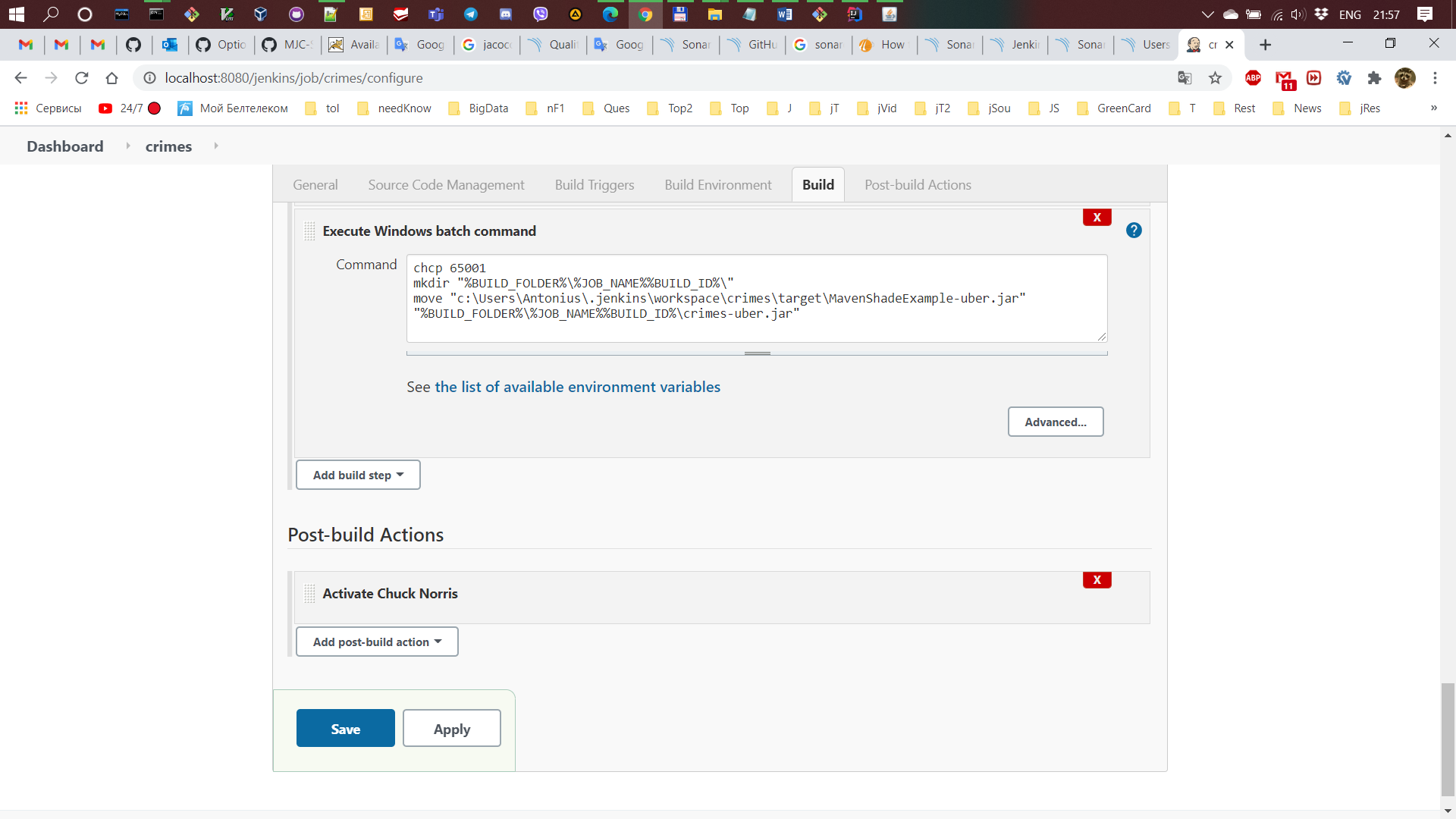


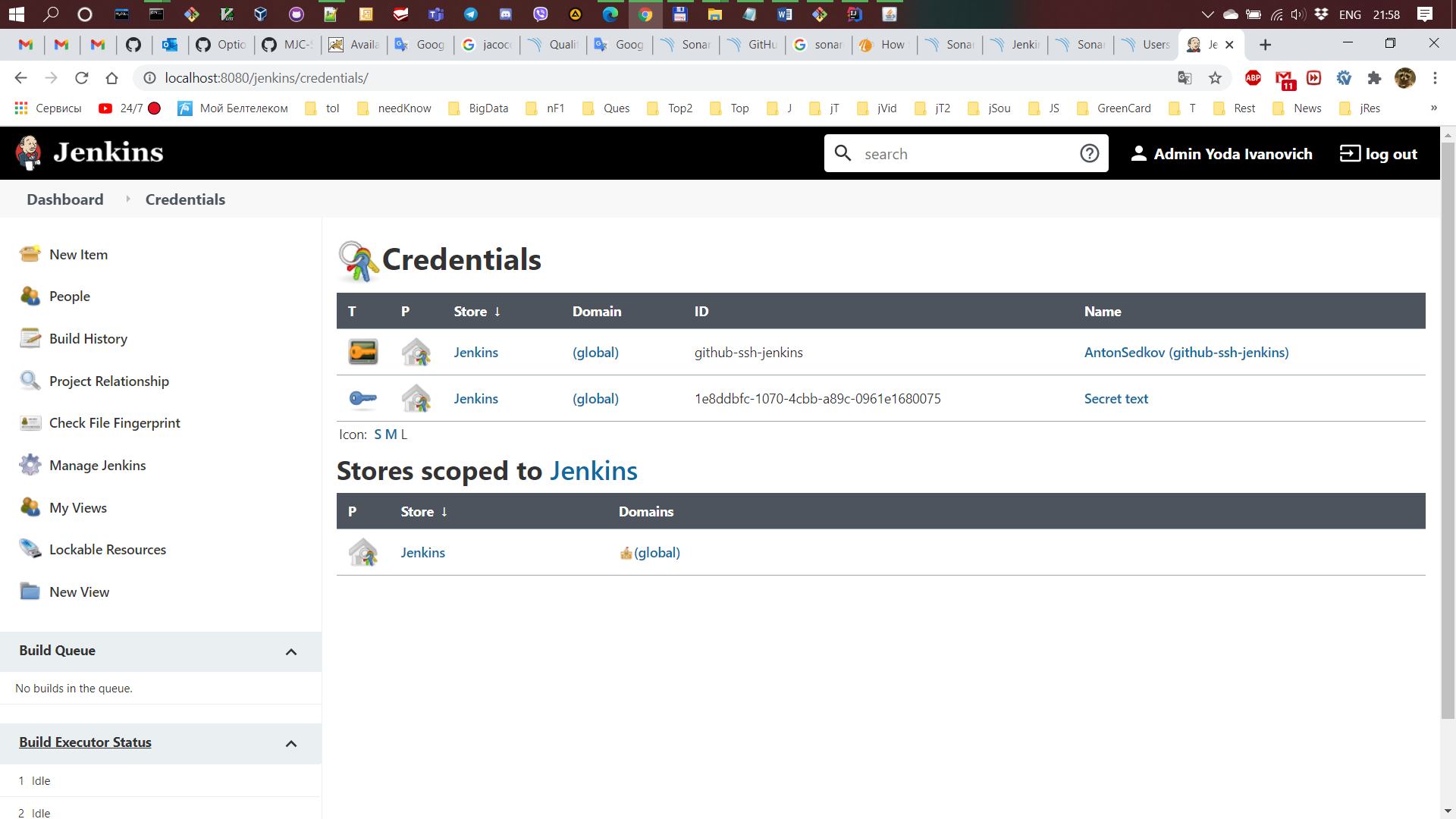


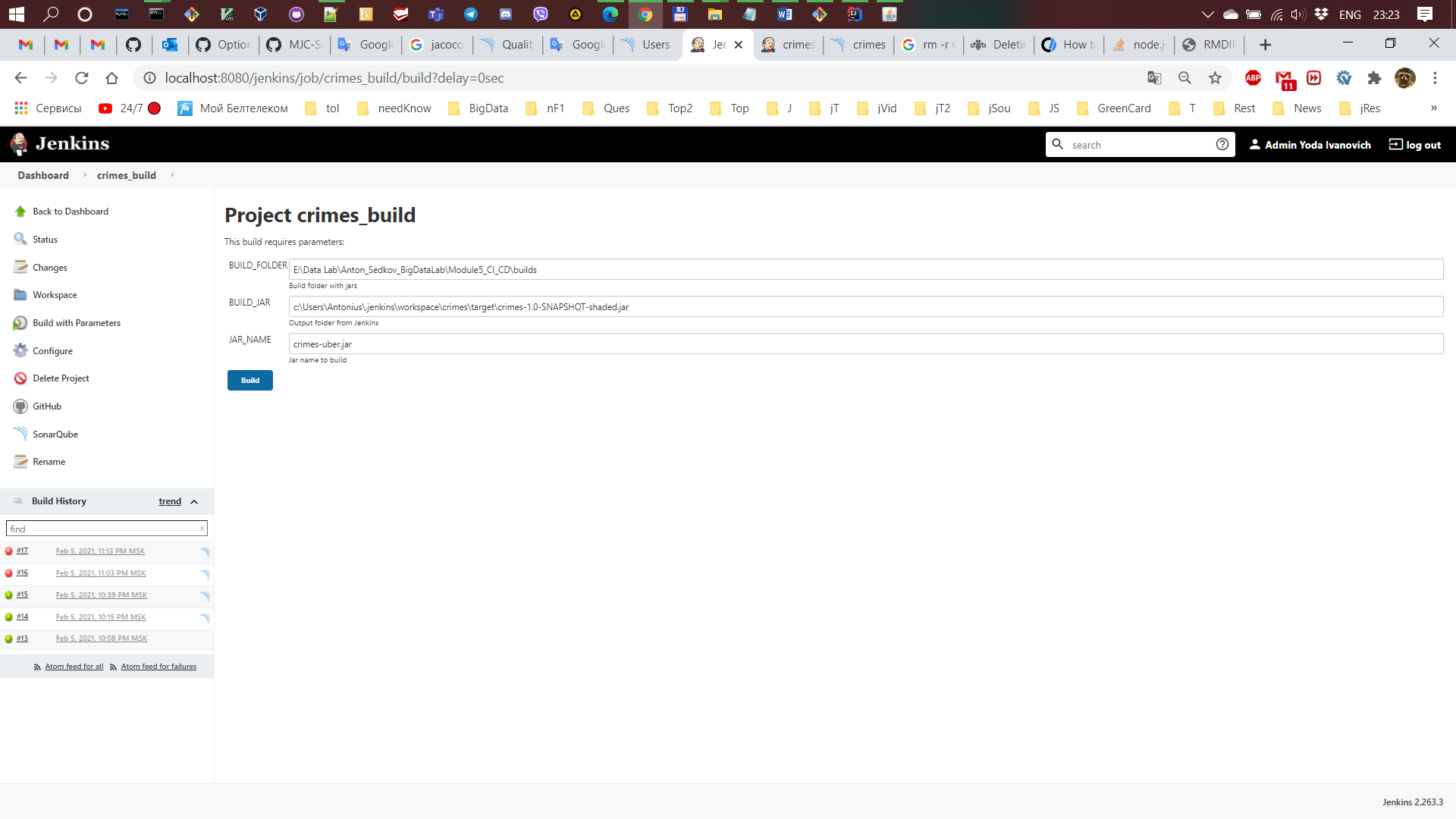


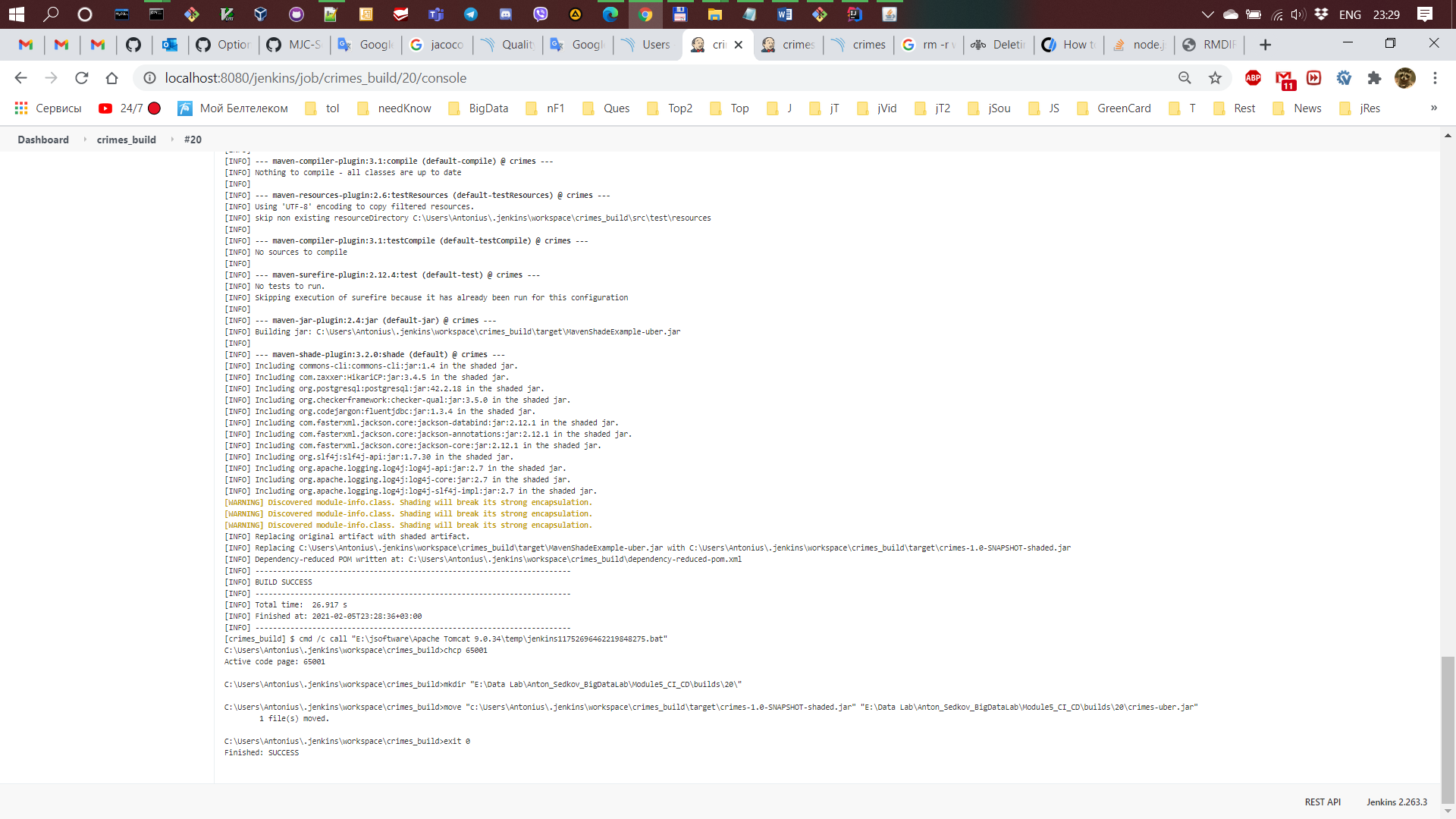




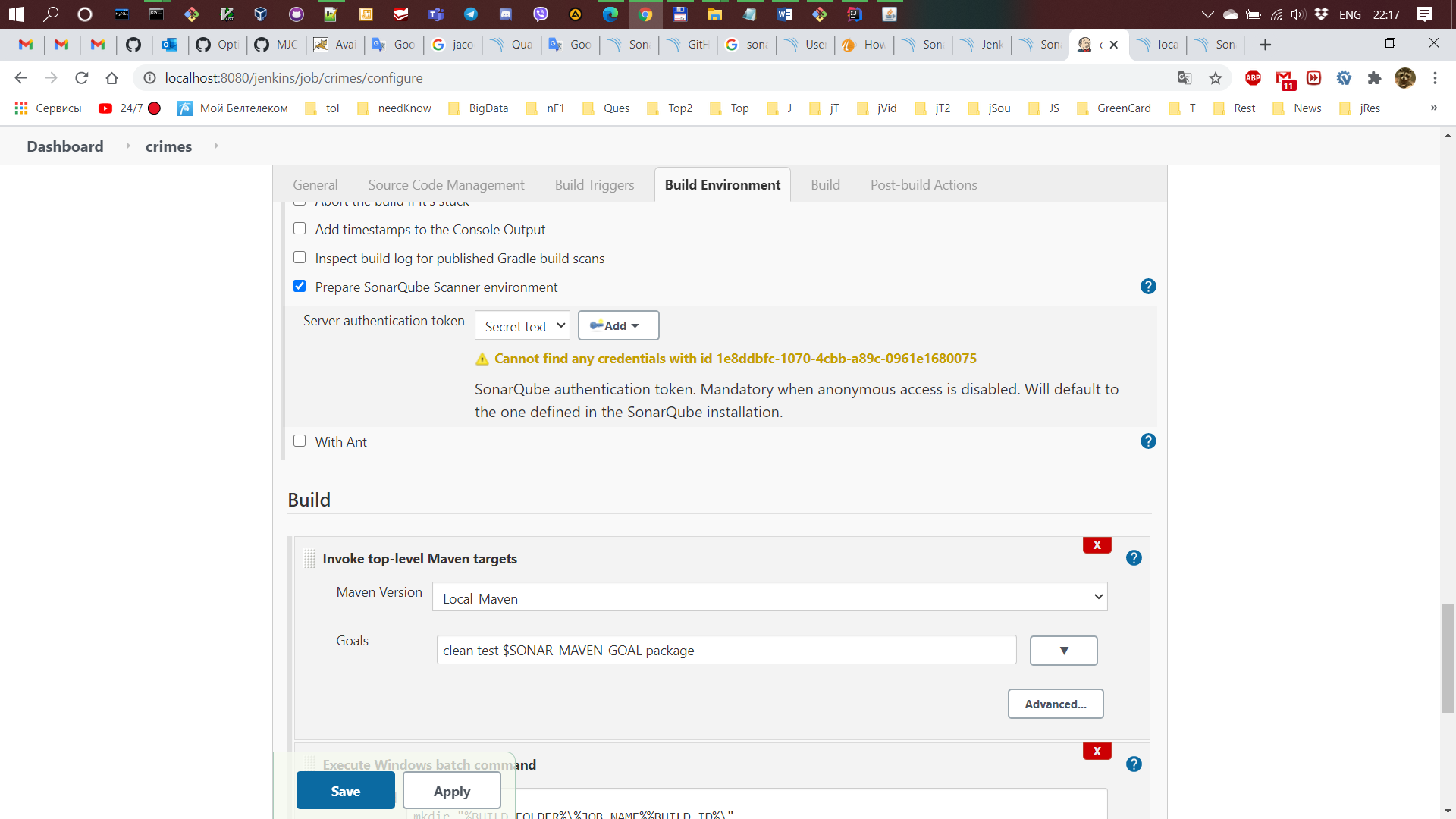


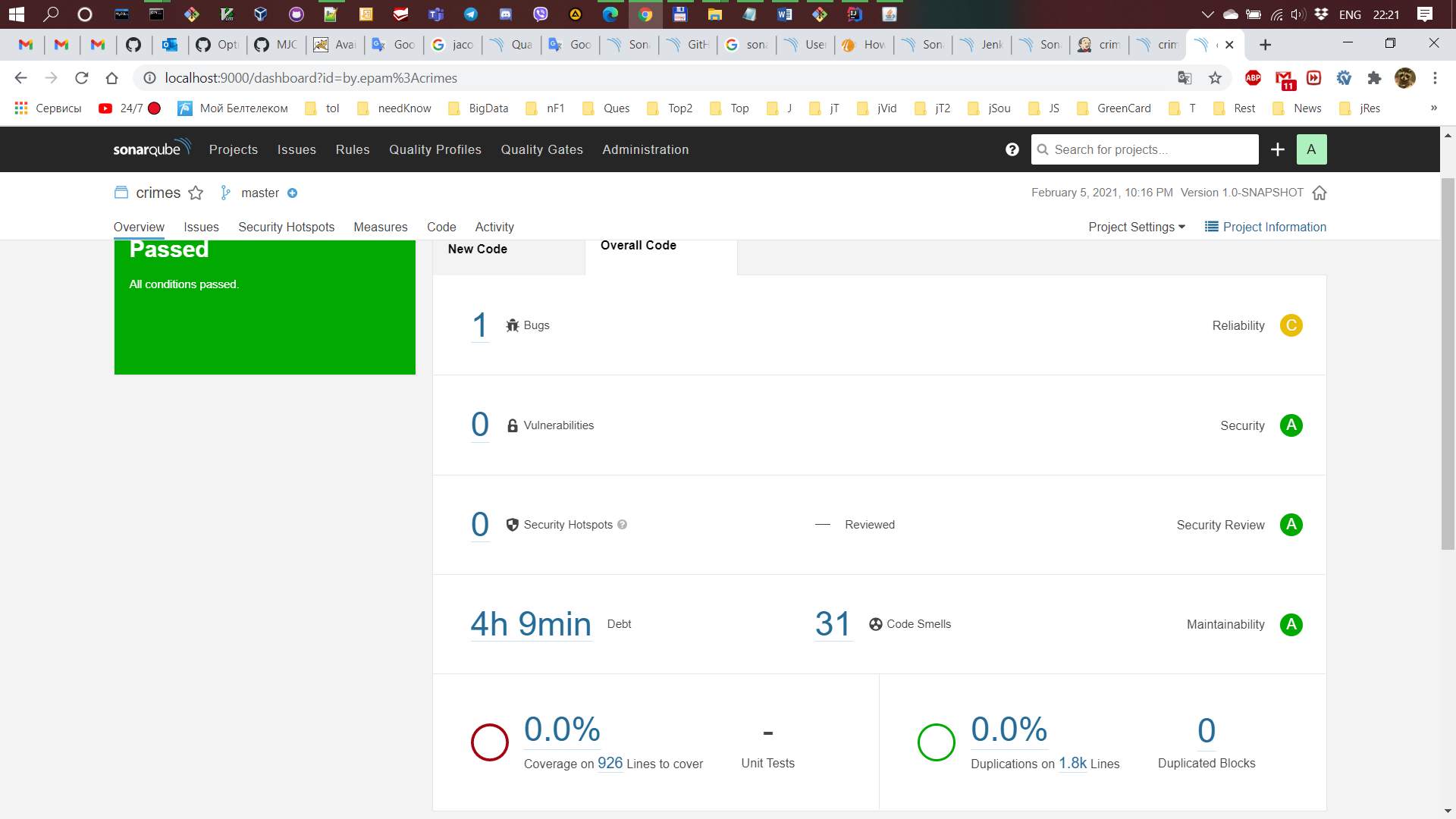


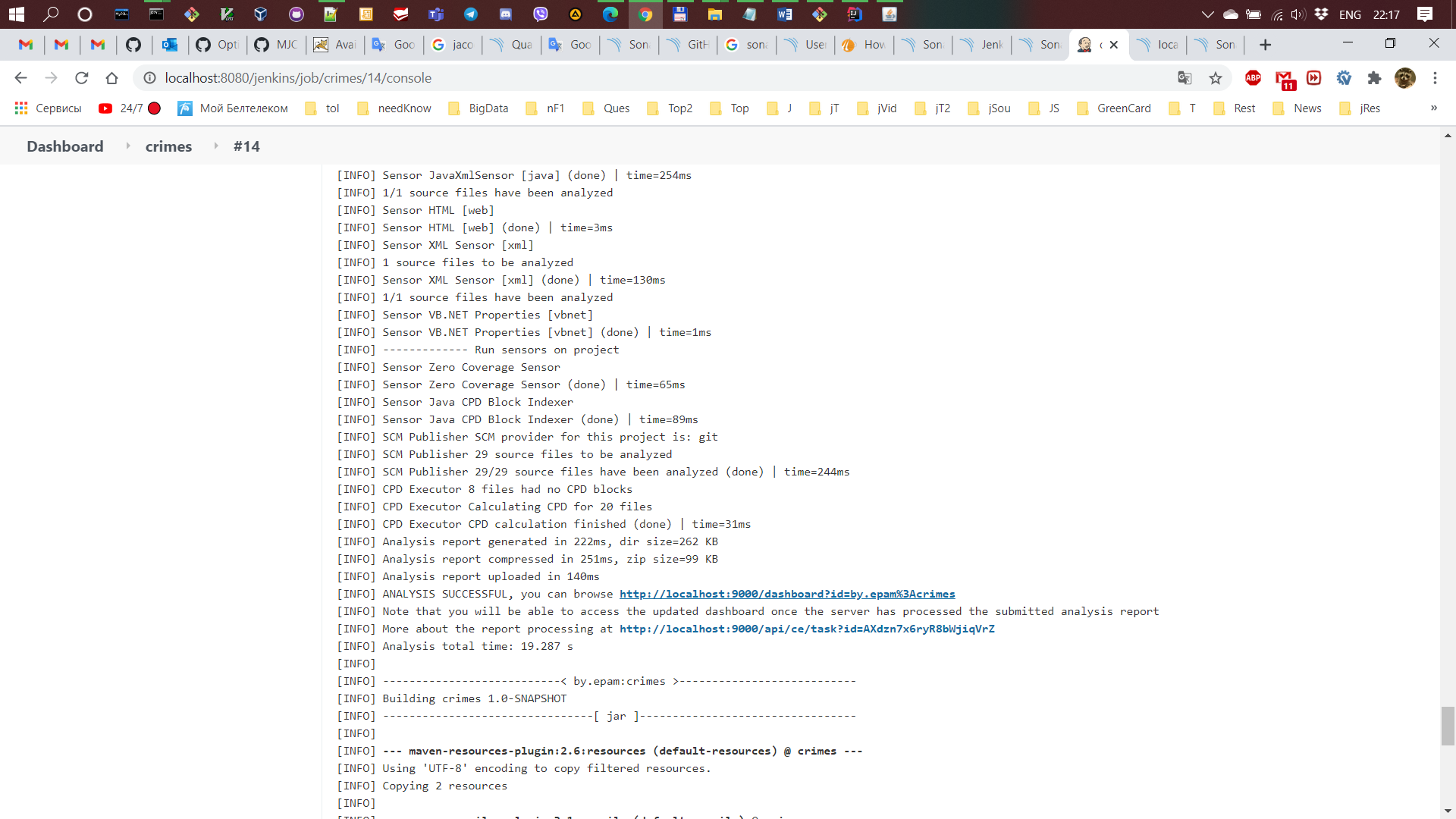




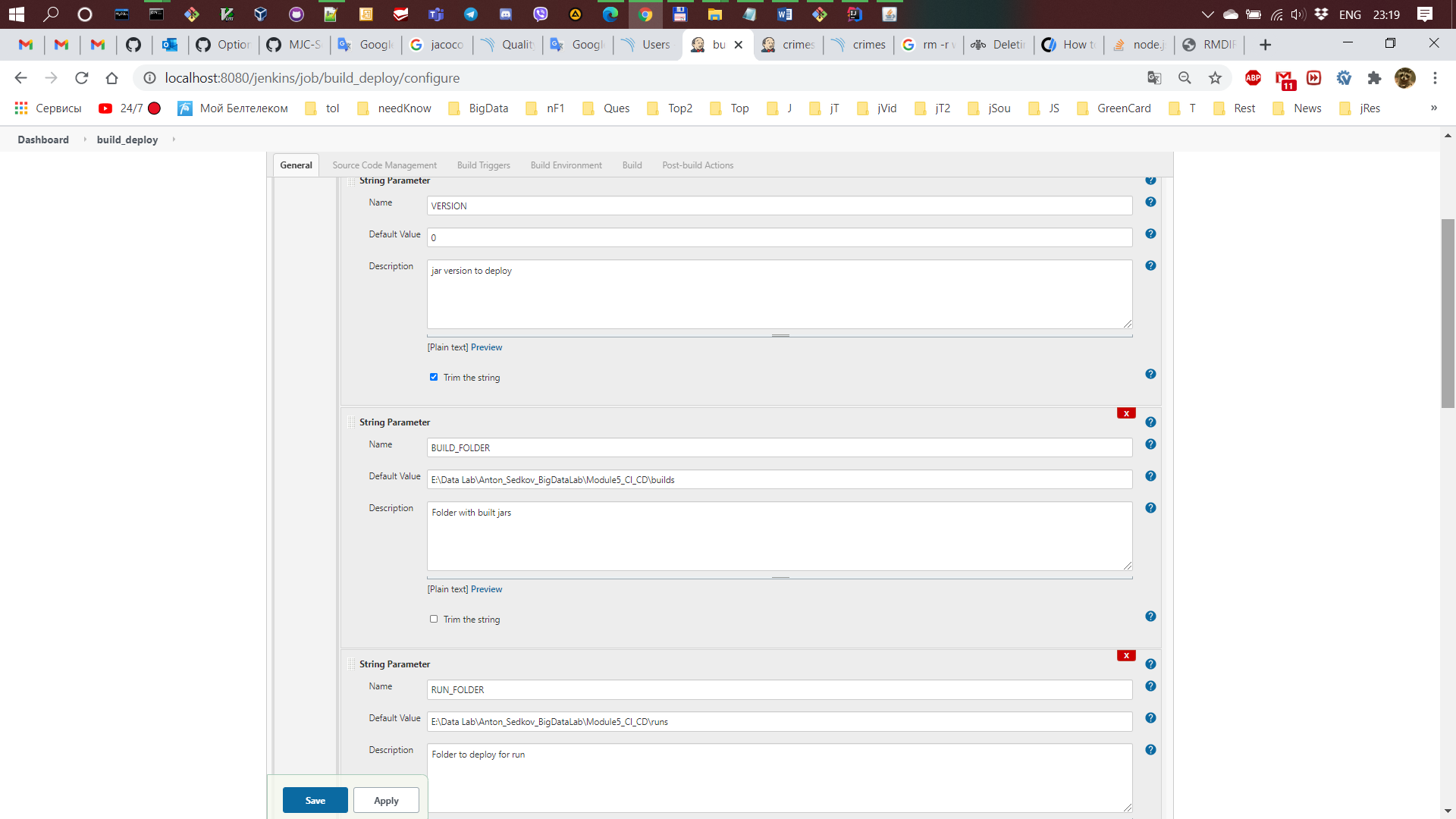
1. Install SonarQube. Configure Jenkins to use local SonarQube installation. In the build job analyze your source code with SonarQube before saving the jar. If the source code does not pass SonarQube quality gate the artifact should not be stored, build job should fail. Use JaCoCo for code coverage.

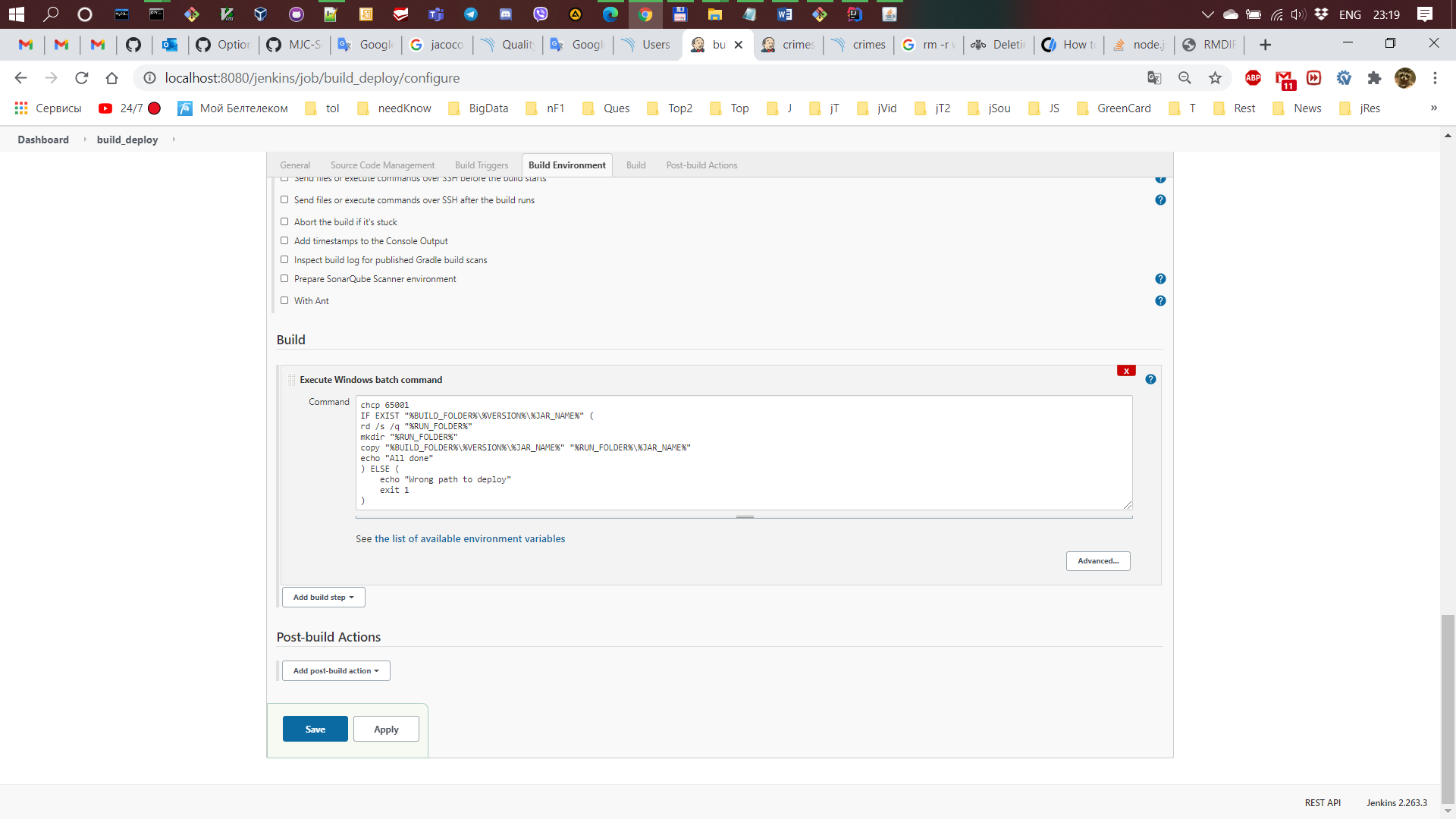


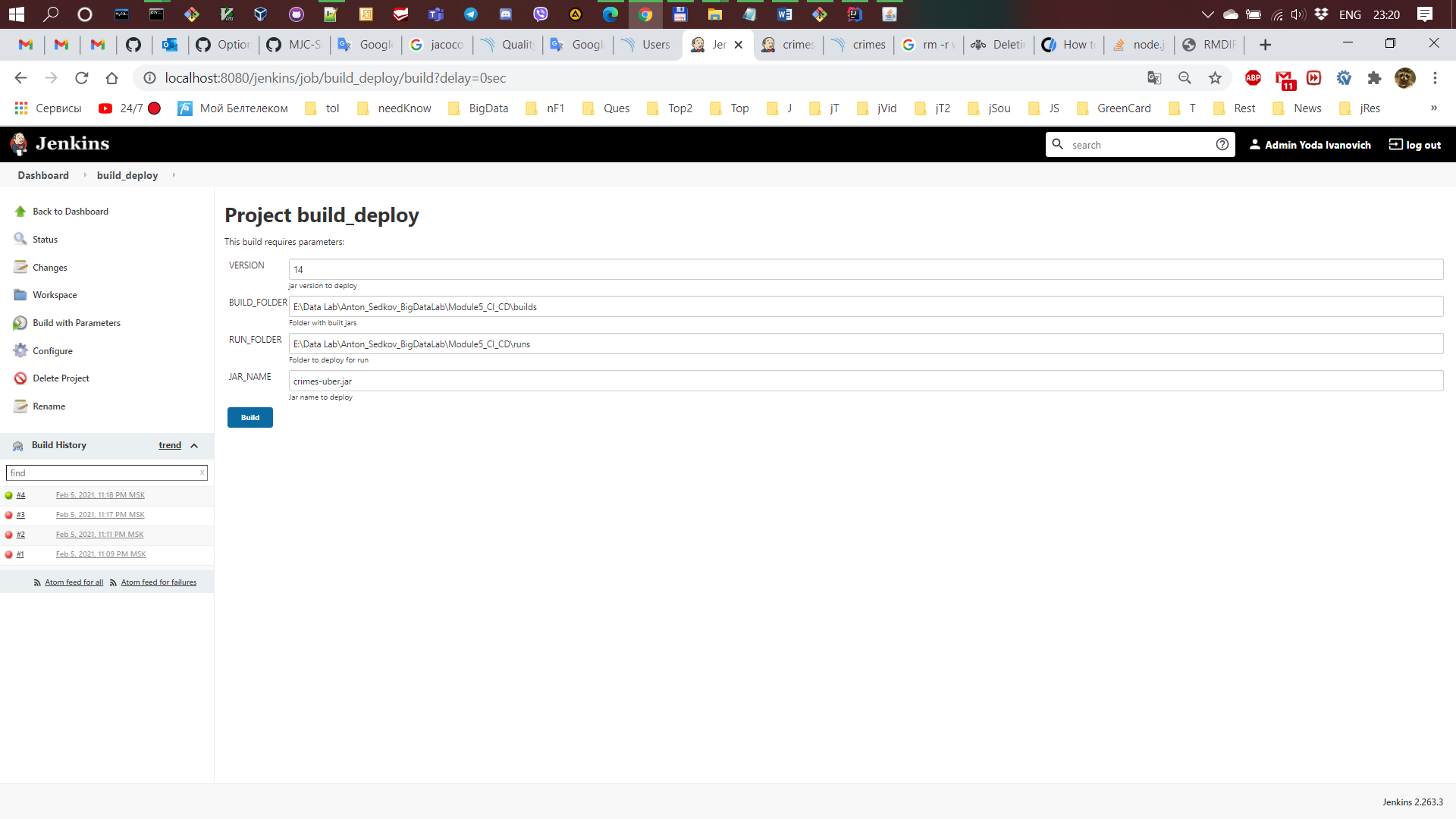


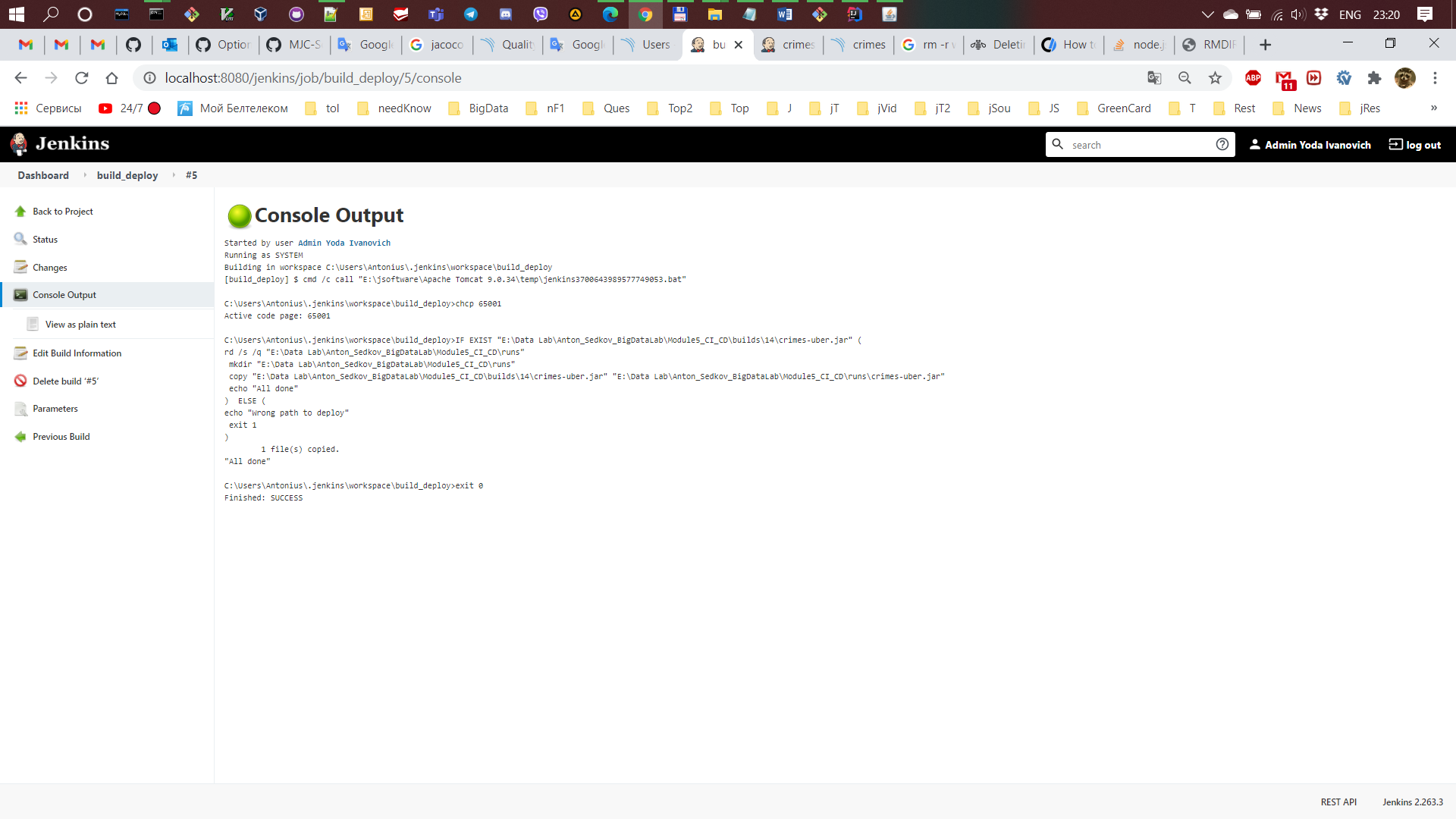


1. Create Jenkins deploy job. Application build version (generated during build job) should be passed as a parameter to the deployment job. Job copies content of the provided build version's folder into the launch folder specified in the job's configuration. Launch folder should be cleaned before deployment.









1. Create Jenkins run job. The job runs application, located in the launch folder, with specified parameters.

