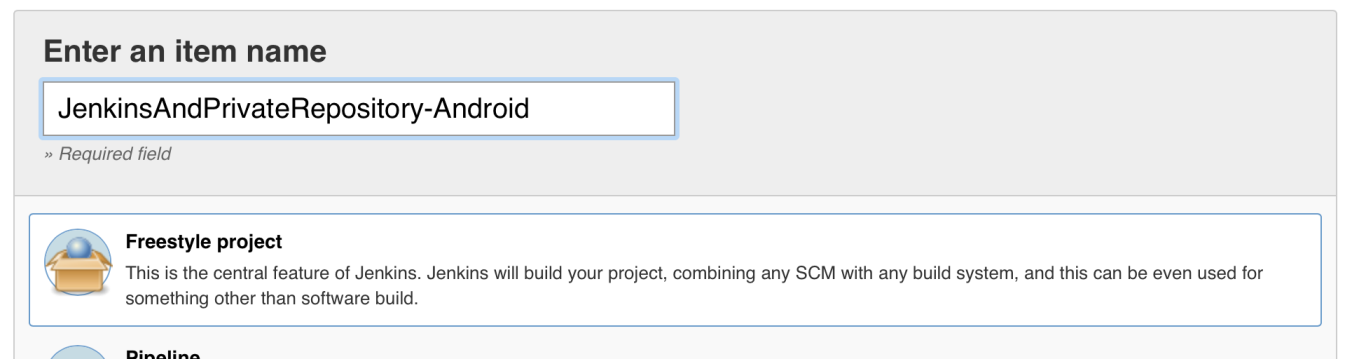
**Create a new Job for Jenkins**

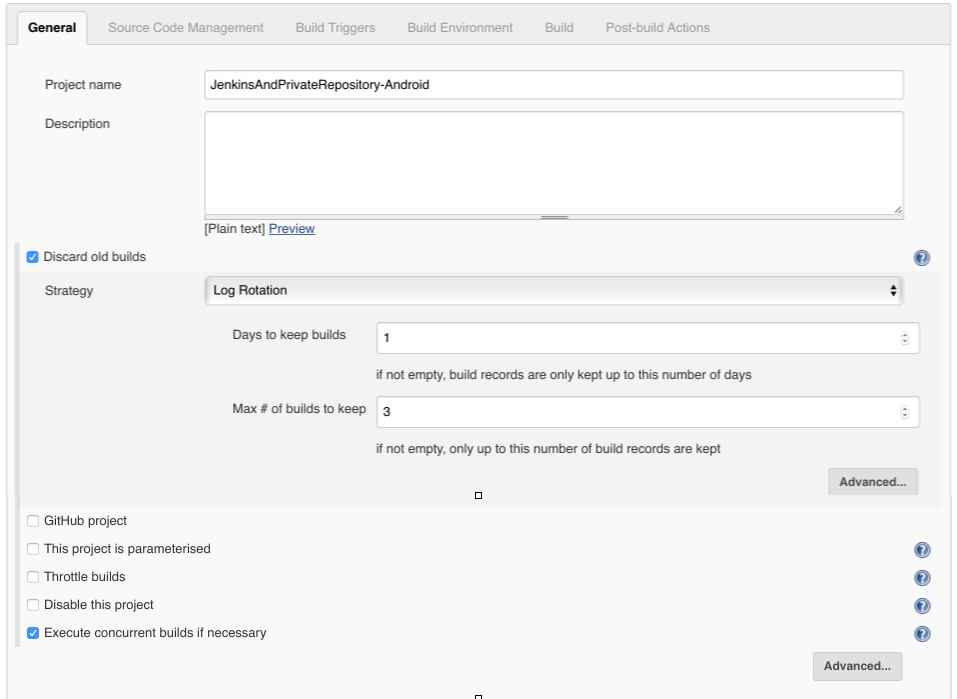
I assume Jenkins completed with all plug-ins installation. It will ask you to create Admin account to Jenkins panel. This account will be your point of control for the panel.

When you’re done and logged in to the panel, click at Create New Job. Name your job whatever you want and choose Freestyle project.



One picture can be more descriptive than hundred words

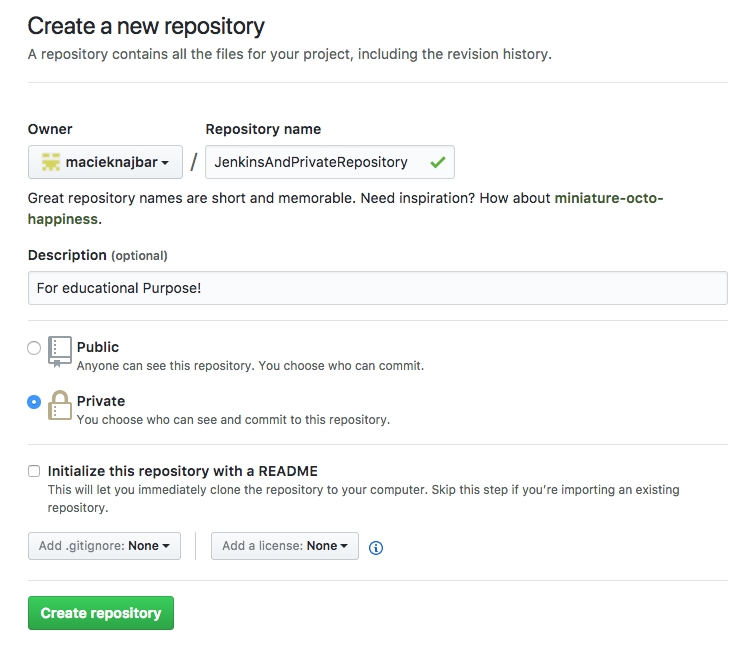
Next step is job configuration. Let’s set only General part. So, we need to set Jenkins to *Discard old builds*(experiment with your own values) and to *Execute concurrent builds if neccessary*.



Save progress.

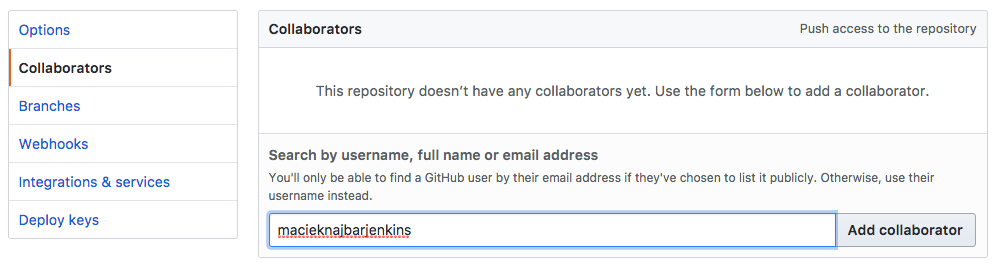
**Create GitHub repository**

Login to your daily GitHub account and create new private repository.



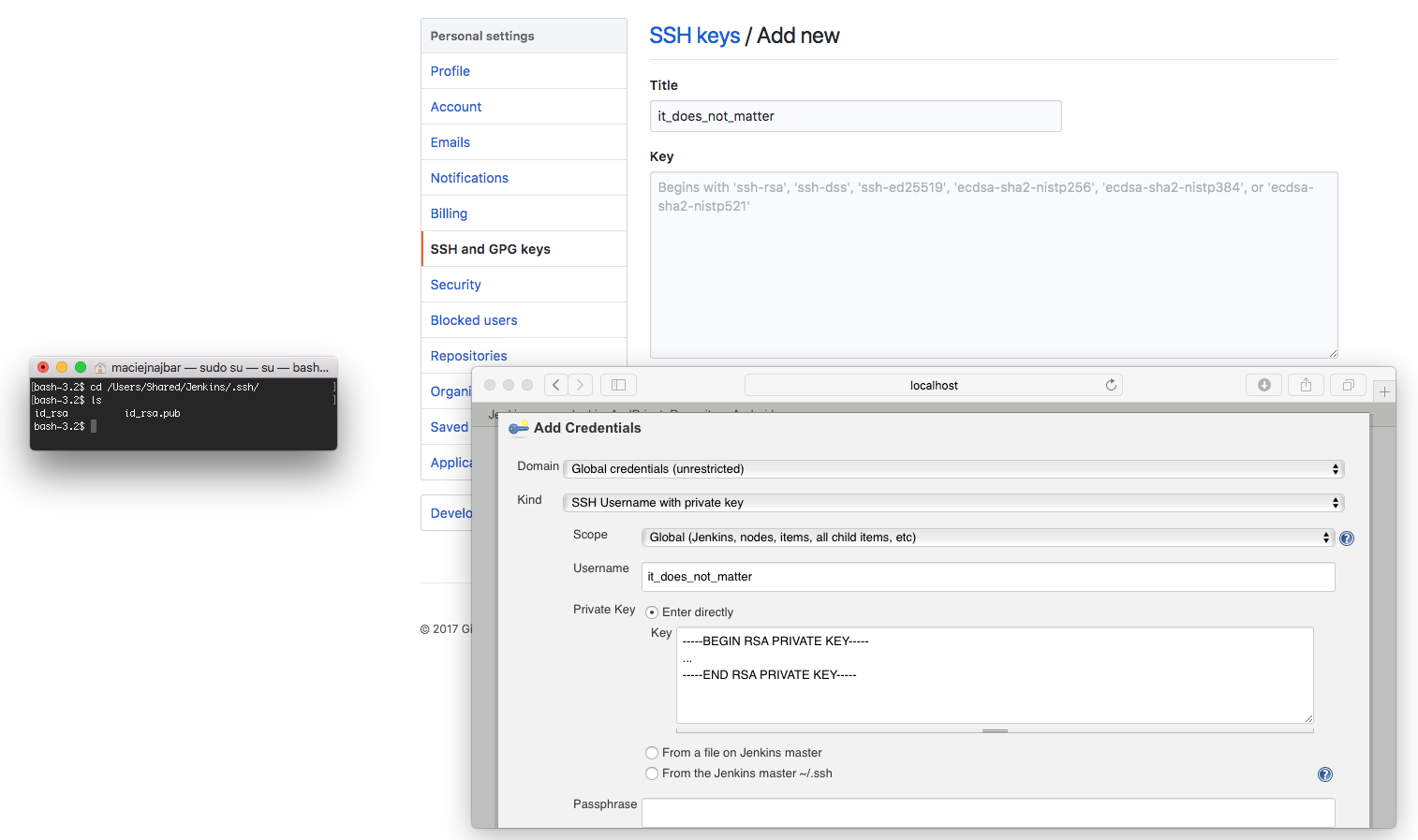
Follow the steps that are displayed to connect your local Android project repository with GitHub remote server.

When it’s done, we need to add Jenkins GitHub account as a collaborator to our project. On the repository ribbon click Settings and go to Collaborators section.



Find your Jenkins account and send an invite. You’ll receive an email to accept an invitation. To accept it, you first need to re-login to Jenkins GitHub account.

Invitation accepted? Awesome! Taking advantage you’re logged in to this account, let’s add SSH key that we previously prepared. Go to your account Settings and enter section *SSH and GPG keys*. Click on *Add new* for SSH keys.



When generating SSH key for Jenkins user, two files were created (id\_rsa and id\_rsa.pub). One of them is public and the other is private. Public key goes to GitHub.

**whoami** *# make sure it says 'jenkins'***cd /Users/Shared/Jenkins/.ssh/  
more id\_rsa.pub**  
(copy the entire output and paste to GitHub)

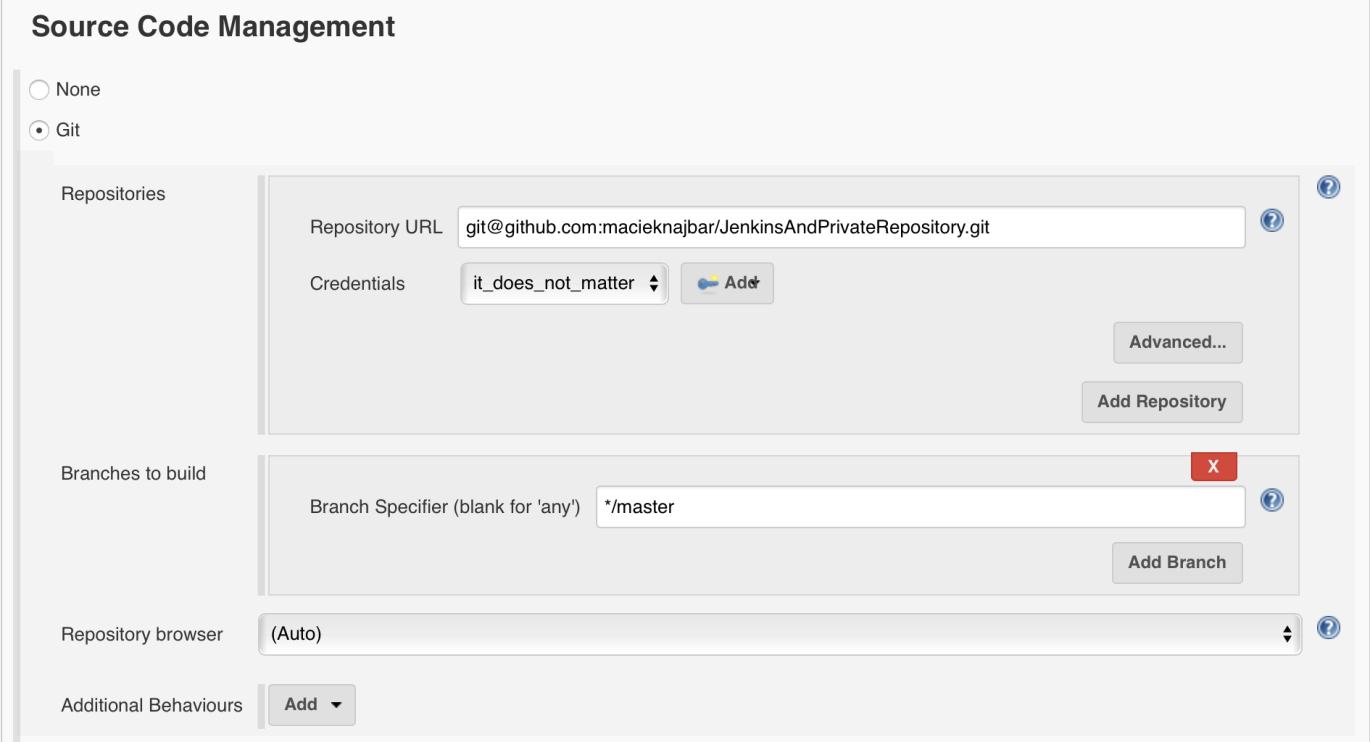
Let’s leave the private key for now. But let’s establish connection for Jenkins to GitHub server. We need to add GitHub to known hosts. Let’s do it by simply connecting to GitHub server.

**whoami** *# make sure it says 'jenkins'***ssh -T git@github.com**  
(And yes, you're sure you want to continue)  
(Result should be a welcome message)

If you got the welcome message it means your Jenkins account finally has access to your repository!

**Continue job configuration**

Let’s connect Jenkins job to our repository. In job configuration go to *Source Code Management* section and choose *Git*.



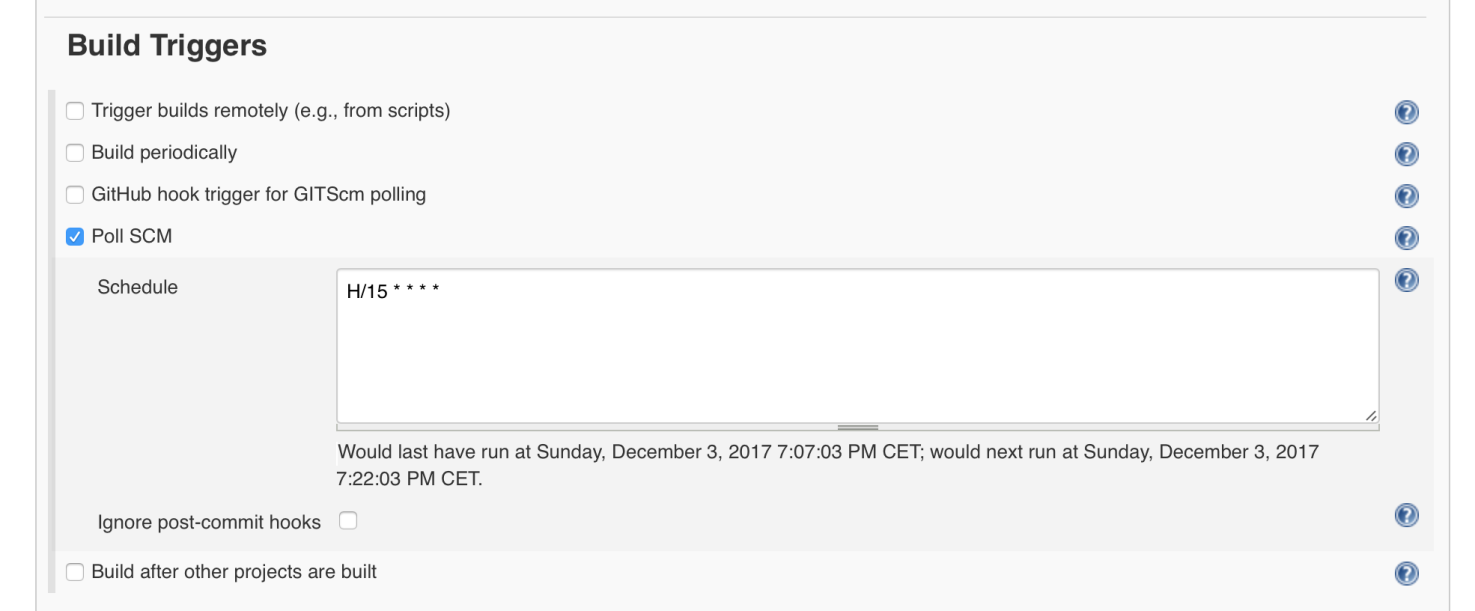
In the picture you can see I have some credentials already, it’s because I created them in the previous picture. Click Add and you’ll see the same screen. Pick Kind: *SSH Username with private key* and type username and again, let’s go to terminal.

**more /Users/Shared/Jenkins/.ssh/id\_rsa** *# This time private key*  
(space, space, space until you see the entire private key)

Copy entire private key as in the picture (including BEGIN and END stuff). Save and choose just created credentials (as in the last screenshot).

Go to Build Triggers and checkoff *Poll SCM*. Schedule it for every 15mins.

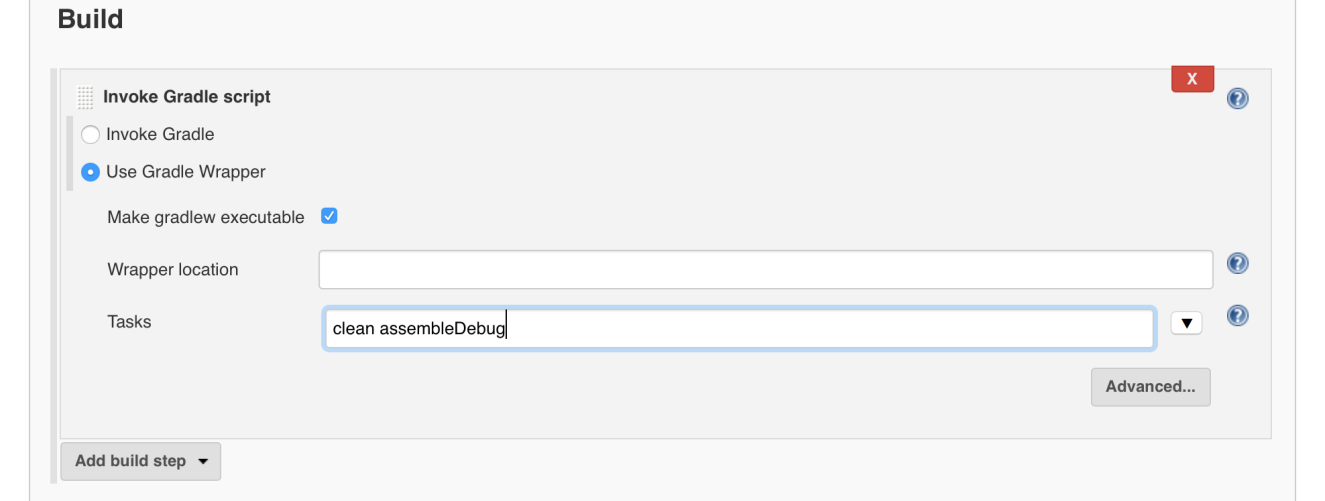
H/15 \* \* \* \*



When you save here and run the build, it will pass. It will connect via SSH to your repository (and that’s basically all it will do).

But we want it to do more. We want it to run our Gradle commands to create build for us. Let’s go back to job configuration view again.

Go to Build section and click *Add build step*. Choose *Gradle script* option and pick *Use Gradle Wrapper* (because this is what Android projects use). Set the tasks you want Gradle to execute.



Save the project and you can run the build again. Works? Excellent!