Continuous deployment of a jenkins job.

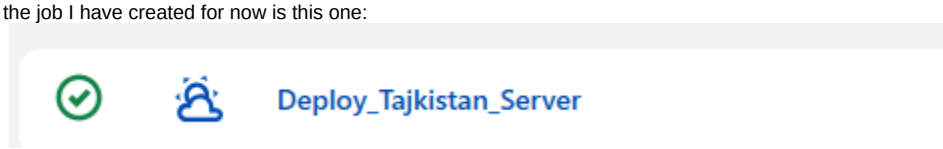
As all jenkins jobs we are using, this is two parts:

1. Creation of the job in jenkins
2. Creating the declarative pipeline in repository jenkinsjobs.

Creation of job in Jenkins

The job created in Jenkins will be stored as an xml document inside the jenkins machine. This xml could be stored in Source Control, but for some reason, it isn't. So take care while modifying jenkins jobs, as there is no backup unless you explicitely ask for a backup of the machine (Alberto Robles).

So the job I have created for now is this one:



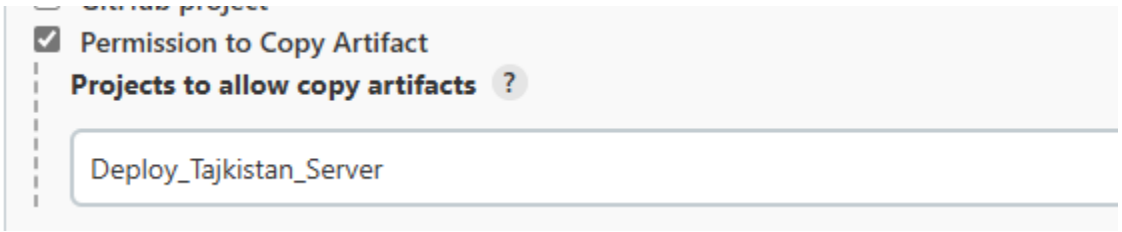
This job will be triggered when the Server Integration one finishes building. The first jobs needs to be archiving their artifacts (Archive Artifact plugin), which was already doing so, and the second job, the one doing the deployment needs to read such artifacts as needed.
To do so, the first job must explicitly say which jobs are allowed to read its archived artifacts.

As a general rule, working with Pipelines in Jenkins is confusing, as there are parts that are defined in the jenkins job, while there are parts defined in the pipeline. This is one of the examples.

So, for the first job, Tajikistan_Server_Git, the archiving part is defined in the pipeline:

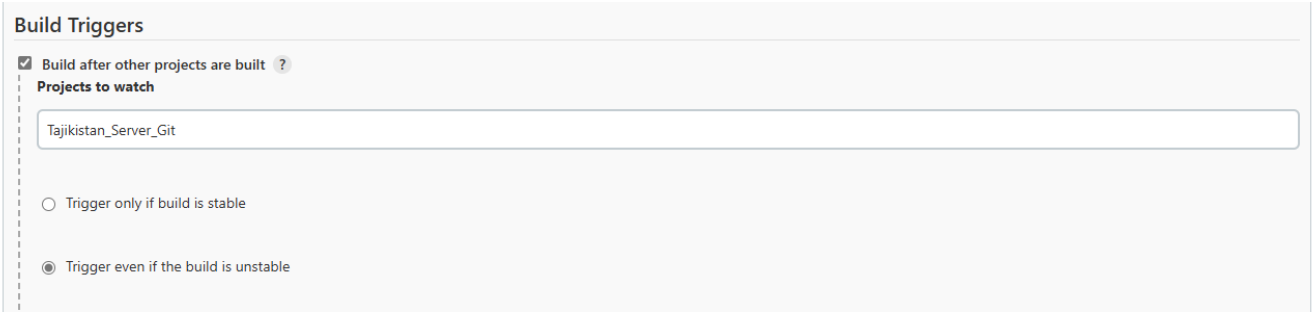
```
success {
    junit 'Server/**/TEST-*.xml'
    archiveArtifacts artifacts: 'Server/Solutions/IMS/build/libs/*, Test/Staging/staging_conf.json, Test/Integration/development_conf.json, Test/Integration/validation_conf.json, Test/Integration/dmz_conf.json', onlyIfSucc
)
```

while defining which projects can read from this, one, is done in the job, in the jenkins ui:



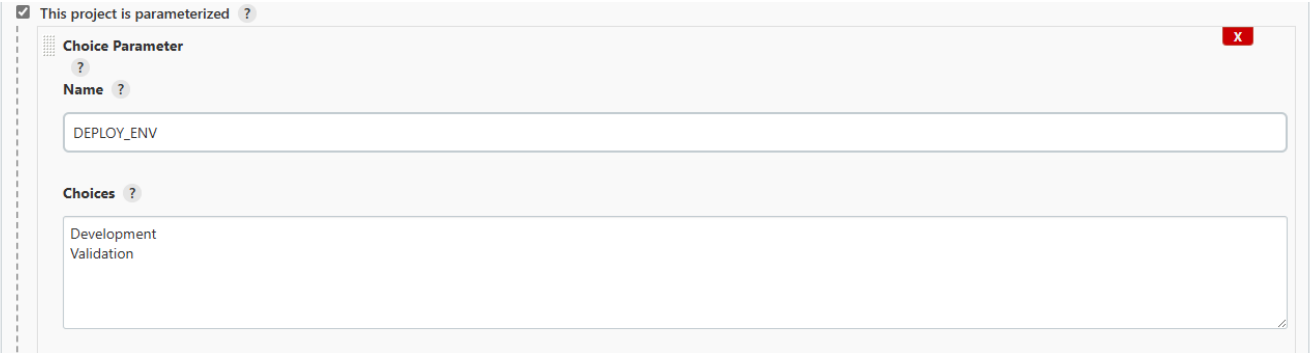
As you can see, we need to define that we give permission to the tailing job to read these artifacts.

In the new job, Deploy_Tajikistan_Server, we need to define some things:
Trigger the build automatically, after the build has finished:



And then, the pipeline we are using:

And if you want, the environment in which to deploy:



Now, this is defined to deploy the last successful build, but that is not defined here, but in the pipeline.

Creating declarative pipeline

Pipeline file is Jenkinsfile_Tajikistan_Deploy_Server.groovy.

Just two steps:

- Copy artifacts.
- Deploy to JBoss.

Copy:

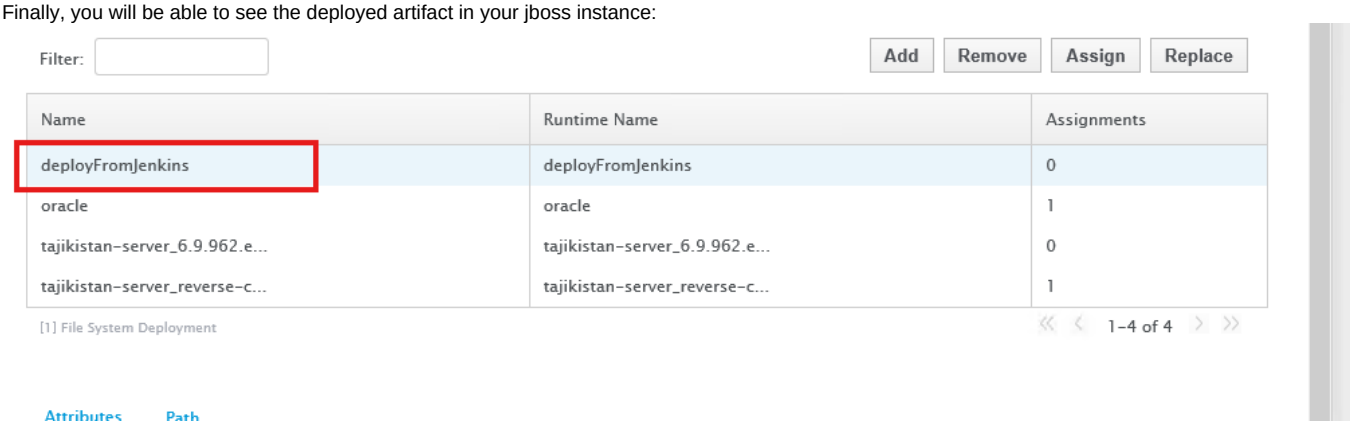
```
steps {
    copyArtifacts(
        projectName: 'Tajikistan_Server_Git',
        selector: [$class: 'StatusBuildSelector', stable:false],
        filter: 'Server/Solutions/IMS/build/libs/tajikistan-server.ear',
        target: '.'
    )
}
}
```

Which build

Deploy into jboss:
Use jboss command line to do so. Also, an auxiliary file for the deployment, as there were too many interpreters to escape:

```
// 3. Deploy into JBoss
def jbossCliCmd = 'C:/Installation/jboss-eap-6.4/bin/jboss-cli.bat --connect --controller=localhost:9999 --file=C:/DeploymentStaging/deploy.cli'
sh """
echo "Complete CLI command: Set-Location C:\\ ; ${jbossCliCmd}"
    ssh -i "$KEYFILE" -o StrictHostKeyChecking=no \\
                      -o PubkeyAcceptedKeyTypes=+ssh-rsa \\
                      -o HostKeyAlgorithms=+ssh-rsa \\
                      -o UserKnownHostsFile=/dev/null \
    ${USER}@${remoteIp} \\
    powershell -Command "${jbossCliCmd}"
"""
```

Finally, you will be able to see the deployed artifact in your jboss instance:



Remote access without password

As the process needs to be non interactive, we need to create an ssh key. Ideally we would create it in the jenkins machine, but as we don't have access we create it locally:

```
ssh-keygen -t rsa -b 2048 -m PEM -f jenkins_deploy_rsa_legacy
```

We are using an old format here because the servers are old.

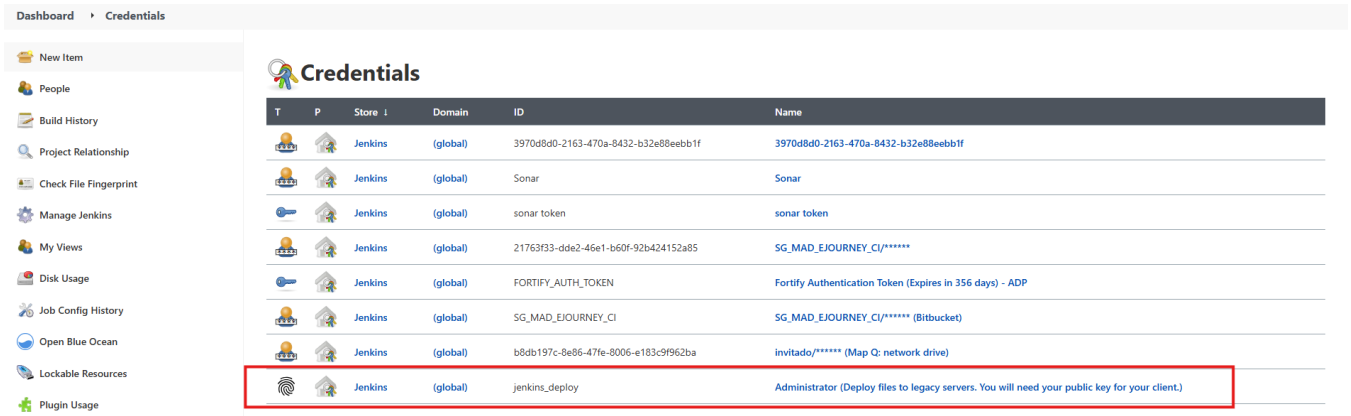
This will create two keys: public and private.

We will use the private key in our jenkins credentials, and the public in our servers, as authroized keys.

Cat your private key and use it as credentials in jenkins:

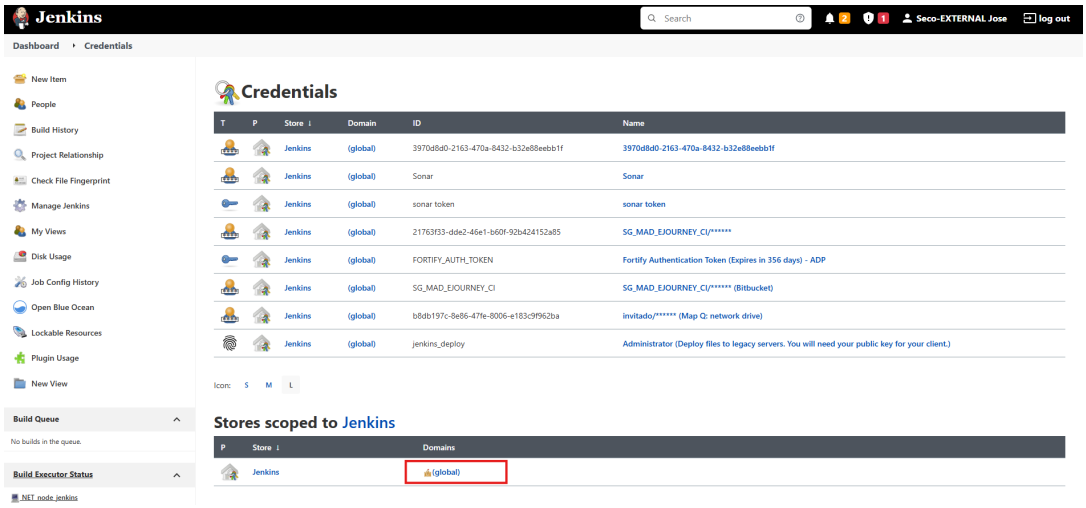
```
$ cat jenkins_deploy_rsa_legacy
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAnZMdEDjgggcDQF/77eZgDQta5uLmIzVz267v2lwxtd/i9tJt
.
.
.
.
.
ClEng57LHHueMdBeZ9jxTzGcXo1U8zY9AlUTtfJzoM9o38rl743h0so=
-----END RSA PRIVATE KEY-----
```

Use it in jenkins, defining the user you want to use: Manage Jenkis Manage Credentials:

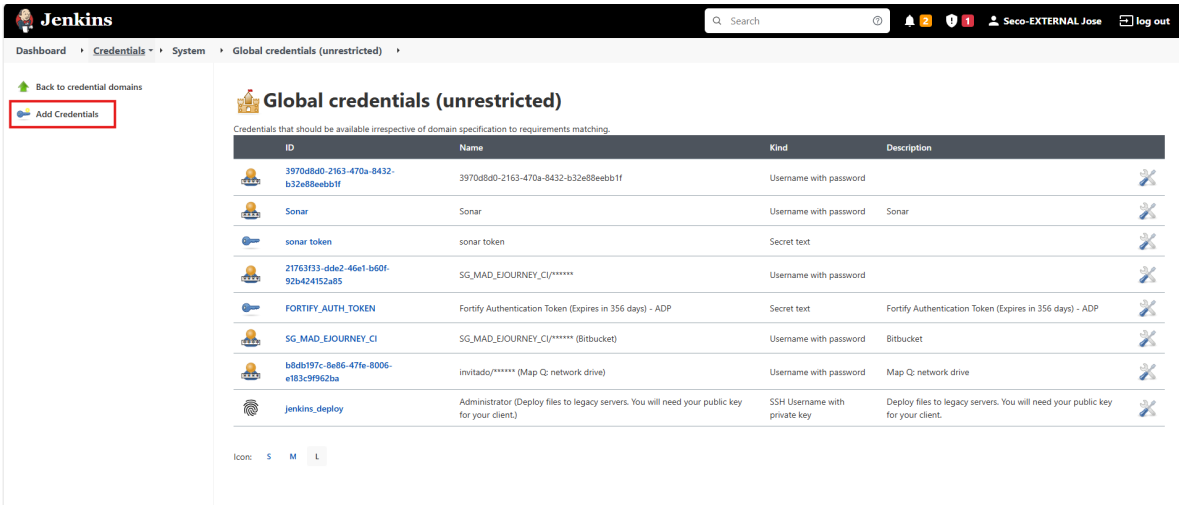


Here Administrator is the user with which to log in our machines.

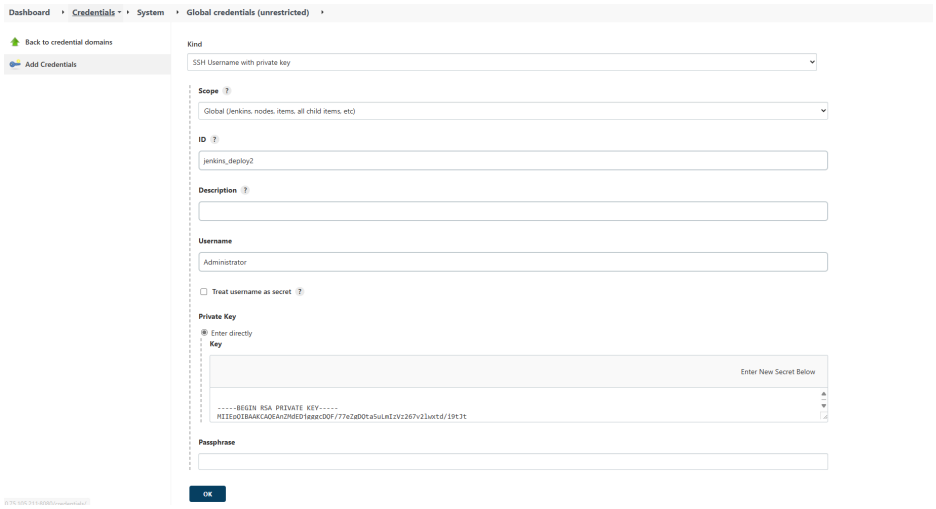
To create a new one, push in the Global Domain:



and then Add Credentials:

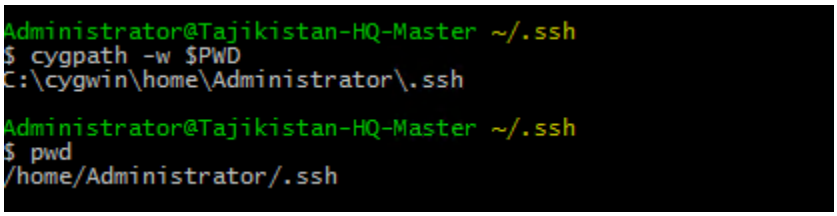


And use your private key there,this is an example:

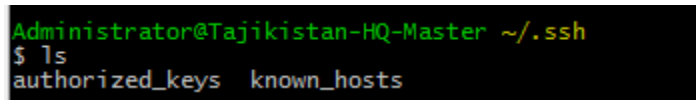


And in the clients (In this case, our clients are the servers where the deployment is to be made), we must have the public keys in the authorized_key files. In our case, we are using the ssh server of cygwin, so the correct file where to place the keys is in /cygdrive/c/Users/Administrator/.ssh/authorized_keys

Here you can see the dir route in linux and windows mode:



Here you need to add your public key (jenkins_deploy_rsa_legacy.pub) in the authorized_keys file:



That way, you will be able to use the ssh commands from the jenkins pipeline without the host asking for user/password.