# Pln-jenkins1 RHEL8 server running Jenkins behind nginx reverse proxy

Use these three Active Directory groups for granting access to users that need to log into Jenkins:

Jenkins Admin - Infrastructure/Platform/DevOps teams

Jenkins QA - QA Team

Jenkins View Only - Developers

# Add Windows Server as Slave Node to Jenkins Master server

1. Jenkins, Manage Jenkins, Manage Nodes and Clouds, New Node
2. Name: pln-qakatalon
3. Check box: Permanent Agent
4. Click ok button
5. Configuration settings of the pln-qakatalon node in Jenkins (print screens):

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. Next screen you will see the commands that need to be ran on the Windows Server that will be the Slave Node (print screen below):
   1. Click on the “agent.jar” link to download that jar file.

A screenshot of a computer

Description automatically generated

1. RDP into the Windows Server as the svc\_pln-qakatalon user
2. Cd to the working directory (which in this case is C:\Users\svc\_pln-qakatalon)
3. Paste the agent.jar file in the root of that working directory
4. Open command prompt as administrator, cd to the working directory
5. Paste the command you received from the Jenkins server earlier

java -jar agent.jar -jnlpUrl https://devjenkins1.americanfirstfinance.com/computer/pln-qakatalon/jenkins-agent.jnlp -secret 812ec8577a119c5124b0a44407b60b71fcc42953b5c3a8261510494b9baf4c5b -workDir "C:\Users\svc\_pln-qakatalon"

1. Connected! (see print screen below)
   1. If you cant connect, ping the Jenkins server (example: ping devjenkins1.americanfirstfinance.com) if you get back an F5 VIP ip address such as 192.168.225.x but you can’t connect then edit the C:\Windows\System32\drivers\etc\hosts file and create a static entry that bypasses the F5 VIP – example:

192.168.207.8 devjenkins1.americanfirstfinance.com

1. Then save the hosts file, ping again and make sure you get the 192.168.207.8 ip. Then run the java command again and it should work!

Text

Description automatically generated

# Add linux slave nodes to Jenkins master server

<https://mohitgoyal.co/2017/02/14/add-linux-slave-node-in-the-jenkins/>

1. On each Jenkins slave node:

sudo yum update

sudo yum install java-11-openjdk

sudo yum install java-11-openjdk-devel

sudo useradd jenkins -U -s /bin/bash

sudo passwd jenkins

* See rob for pwd if needed

vim /etc/sudoers.d/admins

1. add line:

jenkins ALL=(ALL) NOPASSWD: ALL

sudo usermod -aG admins jenkins

1. create Jenkins ssh key one time for use everywhere:

ssh-keygen -t rsa

1. private key name is now jenkins-key on Jenkins master server /var/lib/Jenkins/.ssh/jenkins-key

mkdir .ssh

chmod 0700 .ssh

vim authorized\_keys

1. paste jenkins public key

chmod 0600 authorized\_keys

1. on Jenkins master server, cd .ssh
2. vim config

Host pln-jensl1.aff.local

User jenkins

IdentityFile ~/.ssh/jenkins-key

1. On Jenkins master server, changed /etc/passwd for jenkisn to allow login so can test ssh as jenkins user form this server:

Was: jenkins:x:992:989:Jenkins Automation Server:/var/lib/jenkins:/bin/false

<https://dzone.com/articles/jenkins-03-configure-master-and-slave>

<https://scmquest.com/jenkins-master-slave-setup-and-configuration-with-screenshots/>

1. Jenkins, manage Jenkins, manage plugins
2. Jenkins, manage nodes, new node, select permanent agent
   1. Node name: pln-jensl1
   2. Click ok
3. Description: plano jenkins slave 1 node
4. Remote root directory: /home/jenkins
5. # of executors: 4
6. Launch Method:
   1. Launch agents via ssh
   2. Host: pln-jensl1.aff.local
   3. Credentials:
      1. Add
      2. Jenkins credential provider
      3. Domain: Global credentials (unrestricted)
      4. Kind: ssh username with private key
      5. Scope: global (Jenkins, nodes, items, all child items, etc)
      6. ID: jenkins
      7. Username: jenkins
      8. Private key:
         1. Enter directly:
         2. Click add
         3. Paste in the private key for the jenkins user that we generated earlier
      9. Passphrase: leave blank
      10. Click add
   4. Now select “Jenkins” in the credentials drop down box
7. Node Properties
   1. Check Environment variables:

Name: JAVA\_HOME

Value: /usr/lib/jvm/java-11-openjdk

Name: JRE\_HOME

Value: /usr/lib/jvm/jre-11-openjdk

1. Click save
2. You will see a screen “This node is being launched. See log for more details”
   1. Click on the link for more details
   2. Shows agent successfully connected and online after a few minutes

1. Add another node, node name: pln-jensl2, copy existing node: pln-jensl1
   1. Edit the hostname and description to be the hostname of this new node to connect to: pln-jensl2.aff.local
   2. Make sure no other names need to be updated for host server (shouldn’t need to)
   3. Click save

# PLN-JENKINS1 Ansible deploy server:

*ssh-agent*

*exec ssh-agent bash*

*ssh-add ~/.ssh/id\_rsa*

ansible-playbook -i vms-to-deploy deploy-vsphere-vms.yml

1. add server name to windows dns server
2. if have issue with running playbooks for installs/updates:

*ansible-playbook add-user-ssh.yml -i pln-jenkins1.****aff.local****, -k -K*

hostnamectl

sudo mv /etc/machine-id /etc/machine-id.oldid

sudo systemd-machine-id-setup

sudo subscription-manager register --force

sudo systemctl reboot

ansible-playbook update\_local\_usernames.yml -i *pln-jenkins1.****aff.local***, -K -k

ansible-playbook olawale\_add\_user\_pwd\_and\_ssh\_key.yml -i *pln-jenkins1.****aff.local***, -K -k

root pwd

Reset subscription manager service on server going to deploy to: *sudo subscription-manager register --force*

*ansible-playbook bigfix\_rhel.yml -i pln-jenkins1.****aff.local****,**-K*

*ansible-playbook set-new-created-users-password-expire-90.yml -i pln-jenkins1.****aff.local,*** *-K*

*ansible-playbook jenkins-server-base\_rhel8.yml -i**pln-jenkins1.****aff.local,*** *-K -v*

*ansible-playbook rhel8\_dns\_domain\_pln\_set\_rhel\_ifcfg.yml -i pln-jenkins1.****aff.local,*** *-K*

*~~ansible-playbook install\_composer.yml -i~~* ***~~pln-cdn1.aff.local~~****~~,~~**~~-K~~*

*~~ansible-playbook rhel8\_loggly\_nginx\_local\_solarwinds.yml -i atl-nexus1~~****~~.aff.local,~~*** *~~-K~~*

ansible-playbook nginx\_inserts\_proxylog\_loggly-no-php-fpm.*yml -i pln-jenkins1.****aff.local****,**-K*

*~~ansible-playbook cp\_uat\_rhel8\_appoptics\_nginx\_php\_plugins.yml -i atl-nexus1~~****~~.aff.local,~~*** *~~-K~~*

ansible-playbook nginx\_add\_log\_fwd\_pln\_pln-logfs02\_et.yml *-i pln-jenkins1****.aff.local****,**-K*

*~~ansible-playbook demo8dp\_rhel8\_appoptics\_nginx\_php\_plugins.yml -i~~* ***~~pln-cdn1.aff.local,~~*** *~~-K~~*

*ansible-playbook event\_tracker\_pln-logfs02\_blockinfile\_nginx\_rhel8\_w\_selinux\_fix.yml -i pln-jenkins1****.aff.local,*** *-K*

*ansible-playbook* selinux\_fix\_nginx\_start\_after\_rhel8\_updates.yml *-i pln-jenkins1.aff.local****,*** *-K*

*ansible-playbook* add\_rhel8\_to\_AD\_linux-web-qa\_linux-admin-infra.yml *-i pln-jenkins1****.aff.local,*** *-K*

*ansible-playbook rhel8\_fail2ban.yml -i pln-jenkins1****.aff.local,*** *-K*

*ansible-playbook linux\_local\_solarwinds\_only.yml -I pln-jenkins1.aff.local, -K*

*~~ansible-playbook add\_webrpt\_user\_ssh\_var-log\_ow.yml -i~~* ***~~pln-cdn1.aff.local,~~*** *~~-K~~*

*ansible-playbook logs\_min\_90\_days.yml -i pln-jenkins1****.aff.local,*** *-K*

*ansible-playbook rhel8\_nginx\_show\_realip\_error\_log.yml -i pln-jenkins1.aff.local, -K*

*ansible-playbook ssh\_config\_update\_linux-web-qa\_linux-admin-infra.yml -i pln-jenkins1****.aff.local,*** *-K*

*ansible-playbook sshd\_key\_exchange\_algorithms\_settings\_rhel8.yml -i pln-jenkins1****.aff.local,*** *-K*

*~~ansible-playbook nginx\_modsec\_rule\_949110\_disable.yml -i atl-nexus1.aff.local, -K~~*

8/10/2020: DISABLE MODSEC ON SERVERS EXCEPT DEVAPI FOR NOW

*ALL OTHER SERVERS: ansible-playbook nginx\_modsec\_rule\_949110\_disable.yml -i pln-jenkins1.aff.local, -K*

Reboot server so kernel updates take effect

# Now that server is online and Ansible playbooks ran, we configure the details/configurations on the server:

<https://wiki.jenkins.io/display/JENKINS//Installing+Jenkins+on+Red+Hat+distributions>

<https://computingforgeeks.com/how-to-install-jenkins-on-centos-rhel-8/>

<https://computingforgeeks.com/manage-users-and-roles-in-jenkins/>

<https://docs.cloudbees.com/docs/cloudbees-ci/latest/traditional-install-guide/system-requirements>

$JENKINS\_HOME is /var/lib/Jenkins That is where everything is stored

1. Added second hard drive, 700GB, to vm for jenkinshome storage
2. In vmware
   1. Pln-jenkins1 server
   2. Edit settings
   3. Add New Device
   4. Hard Disk
   5. 700GB
   6. Click OK

ssh into server

sudo rescan-scsi-bus.sh

lsblk

1. new drive is sdb

fdisk /dev/sdb

1. command prompt enter n for new partition:

n

partition type (primary): p

partition number: 1

1. use default 2048 for first sector
2. for last sector just press enter to use entire drive
3. command: w to write your changes

umount /dev/sdb1

mkfs.xfs /dev/sdb1

mkdir /opt/jenkinshome1

mount /dev/sdb1 /opt/jenkinshome1

mount | grep /dev/sdb1

lsblk -f

copy uuid: 17966506-cf38-49c1-baa5-7ea11074c548

└─sdb1 xfs 17966506-cf38-49c1-baa5-7ea11074c548 /opt/jenkinshome1

vim /etc/fstab

1. paste this in for your uuid and mount point:

UUID=17966506-cf38-49c1-baa5-7ea11074c548 /opt/jenkinshome1 xfs defaults 1 1

1. Reboot server and make sure everything comes back up (boot doesn’t hang) and drive /opt/jenkinshome1 automatically happens
2. Everything looks good!

systemctl stop jenkins

cd /var/lib/jenkins

sudo mv .\* /opt/jenkinshome1/

ln -s /opt/jenkinshome1/ /var/lib/jenkins

1. now /var/lib/jenkins will link to /opt/jenkinshome1
2. make sure permissions are correct by reapplying them:

sudo chown -R jenkins:jenkins /opt/jenkinshome1

1. start Jenkins service

su - jenkins

systemctl start Jenkins

1. log into Jenkins – all is well!

cat /var/lib/jenkins/secrets/initialAdminPassword

/var/lib/jenkins/config.xml

/etc/nginx/nginx.conf

*sudo dnf module list maven*

*sudo dnf module enable maven:3.6/common*

sudo dnf install maven

<https://jenkins1.americanfirstfinance.com>

1. click install suggested plugins (below were the auto selected/installed with this):
   1. folders
   2. timestamper
   3. pipeline
   4. git
   5. ldap
   6. owasp markup formatter
   7. workspace cleanup
   8. github branch source
   9. ssh build agents
   10. email extension
   11. build timeout
   12. ant
   13. pipeline: github groovy libraries
   14. matrix authorization strategy
   15. mailer
   16. few others I missed before screen changed
2. prompted to create first admin user:
   1. created local accounts until Raul can get LDAP integration enabled:
   2. local users: rob, olawale, raul
3. Jenkins url: <https://jenkins1.americanfirstfinance.com>
4. After logged into Jenkins went to Plugin Manager and added:
   1. Bitbucket
      1. Integrates with BitBucket
   2. Bitbucket Push and Pull Request
      1. Integrates with Bitbucket Cloud (rest api version >=2.0) Server triggering on push and pull requests.
   3. Blue Ocean
   4. Health Advisor by CloudBees
   5. JIRA Integration for Blue Ocean
   6. JIRA Trigger
   7. Maven Integration
   8. OpenShift Sync
      1. Sync your OpenShift BuildConfigs with Jenkins jobs.
   9. OpenShift Client
      1. This plugin provides Jenkins pipeline DSL interactions for OpenShift.
   10. OpenShift Login
       1. Allows you to log in to Jenkins via OAuth to an OpenShift installation
   11. Nexus Artifact Uploader
   12. Nexus Platform
   13. Slack Notifications
   14. Download now and install after restart

## Create Bitbucket user for Jenkins to use to connect to Bitbucket

1. Log into bitbucket cloud
2. AmericanFirstFinance space
3. Members, Manage Users, Add Member
   1. Email address: [~~rbruner@americanfirstfinance.com~~](mailto:rbruner@americanfirstfinance.com) jenkins@americanfirstfinance.com
4. Groups: administrators
5. Check your email to accept
6. Full name: ~~svc\_jenkinsaff~~ Jenkins
7. Pwd:
8. Username: svc\_jenkinsaff
9. Member name:
   1. Took 24 hours for it to update from User 60933 to Jenkins on Atlassian profile. And 24 hours before profile picture could be updated.
10. Once logged into bitbucket as this new user svc\_jenkinsaff, click SSH keys
11. Added public ssh key I generated on jenkins1 server

ssh-keygen

/home/rbrunner/svc\_jenkins/id\_rsa

1. Jenkins, Credentials, System, Global credentials (unrestricted), svc\_jenkins

## Need Docker installed on the Jenkins server to run build jobs in different environments

yum config-manager --disable docker-ce-stable

sudo dnf install https://download.docker.com/linux/centos/7/x86\_64/stable/Packages/containerd.io-1.2.13-3.2.el7.x86\_64.rpm

sudo yum list | grep containerd.io

sudo dnf install https://download.docker.com/linux/centos/7/x86\_64/stable/Packages/docker-ce-cli-19.03.13-3.el7.x86\_64.rpm

sudo dnf install https://download.docker.com/linux/centos/7/x86\_64/stable/Packages/docker-ce-19.03.13-3.el7.x86\_64.rpm

1. verify the latest docker-ce version is now installed:

sudo dnf install docker-ce

sudo systemctl enable docker.service

sudo groupadd docker

sudo usermod -aG docker rob

sudo usermod -aG docker jenkins

sudo systemctl start docker.service

## Jenkins Active Directory plugin – integration to allow multiple groups

Jenkins recognizes all the groups in Active Directory that the user belongs to, so you can use those to make authorization decisions (for example you can choose the matrix-based security as the authorization strategy and perhaps allow "Domain Admins" to administer Jenkins).

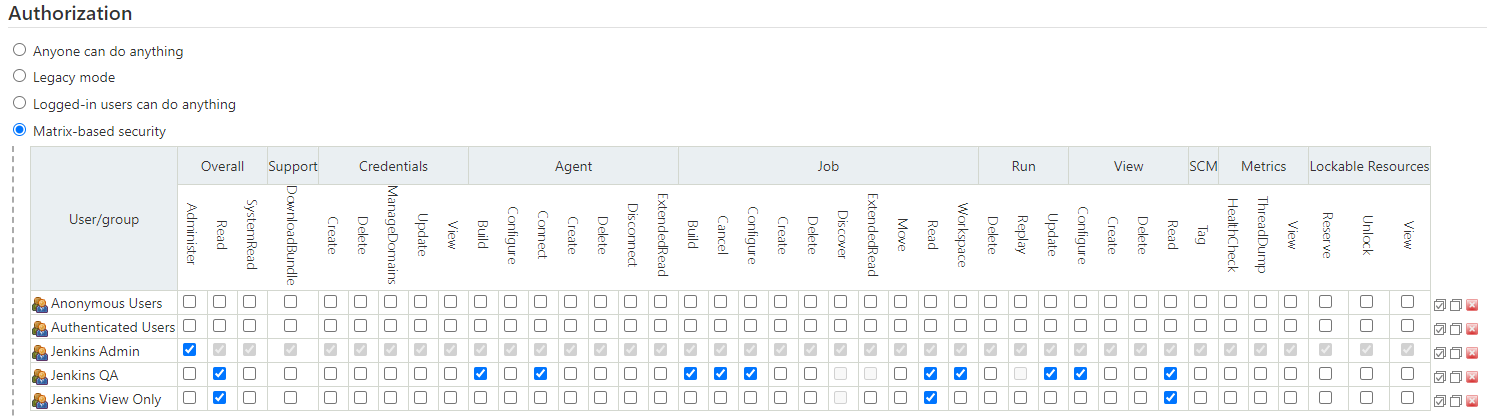
<https://plugins.jenkins.io/active-directory/>

Test server to try changes in: <https://qajenkins1.americanfirstfinance.com/>

<https://medium.com/modern-stack/implementing-active-directory-based-security-in-jenkins-f78dbac929de>

<https://www.thegeekstuff.com/2016/06/jenkins-security/>

1. Manage Jenkins, Manage Plugins, add Active Directory plugin 2.23
2. Manage Jenkins, Configure Global Security, Select Active Directory, Add Domain
   1. Domain name: aff.local
   2. Domain controller:
      1. For ATL Jenkins server: 192.168.149.20
      2. For PLN Jenkins server: 192.168.211.20
   3. Bind DN: aff\svc\_jenkinlsldap
      1. Bind Password:
   4. Bind Password: enter password for this service account user to talk to AD
   5. Test and apply changes
3. You can log out then login using your AD username/password
4. **Configuring Role Based Security**
5. Now that we have configured Jenkins to use AD Integration, we can now take that configuration a step further and manage our users permissions, access levels, and rights by leveraging the AD\Windows Groups.
   1. Add the three groups under Matrix-based security:
      1. Jenkins View Only:
         1. Overall section gets Read check box
      2. Jenkins QA:
         1. Overall section gets Read
         2. Job section gets:
            1. Build, Cancel, Configure, Read, Workspace
      3. Jenkins Admin: Overall Administer (this is the administrators group for Jenkins)



<https://sqlnotesfromtheunderground.wordpress.com/2017/01/27/connecting-jenkins-to-active-directory-and-how-to-reset-access-to-jenkins/>

## ~~Setup LDAP~~

1. ~~Configure Global Security~~
2. ~~Security Realm: LDAP~~
3. ~~Server: ldap://aff.local~~
4. ~~Root DN: OU=AFF,DC=aff,DC=local~~
5. ~~User search filter: (&(sAMAccountName={0})(memberOf=CN=Access-Web Jenkins,OU=Groups,OU=AFF,DC=aff,DC=local))~~
6. ~~Group search filter: (& (cn={0}) (objectclass=group) )~~
7. ~~Group membership: Search for LDAP groups container user~~
8. ~~Manager DN: CN=svc\_jenkinlsldap,OU=ServiceAccounts,OU=Users,OU=AFF,DC=aff,DC=local~~
9. ~~Manager password: concealed (pwd for the AD service account)~~
10. ~~Display Name: displayname~~
11. ~~Email Address LDAP attribute: mail~~

## Setup Jenkins CI integration to Slack:

<https://americanfirstfinance.slack.com/services/new/jenkins-ci>

<https://americanfirstfinance.slack.com/services/B01FKK5R5GT?added=1>

<https://kunzleigh.com/creating-a-slack-notifier-using-jenkins-pipeline/>

<https://www.jenkins.io/blog/2017/02/15/declarative-notifications/>

#jenkins

Slack plugin, in Jenkins:

1. Configuration:
2. Wordspace: americanfirstfinance
3. Credential: Create a secret text
   1. Scope Global
   2. Secret: enter the token provided
   3. ID: slack-token
   4. Description: Jenkins slack channel post
4. Default channel / member id: Jenkins
5. Save
6. In Jenkins click Test button
7. For each Project that you would like receive notifications for, choose Configure from the project's menu.
8. Then you'll need to add Slack Notifications to the Post-build Actions for this project.
9. On multi-configuration project can select Post-Build Actions, Post to Slack, check all

Jenkins home when Jenkins user is service account. Use this method to install SSH keys that jenkins server should have access to at the CLI level for ssh / rsync

<https://stackoverflow.com/questions/47446724/jenkins-pipline-rsync-some-files-error>

1. can still login as Jenkins user even though it doesn’t have home dir or its set to false:

sudo su - jenkins -s/bin/bash

whoami

echo $HOME

-bash-4.1$ cd .ssh

-bash-4.1$ pwd

/var/lib/jenkins/.ssh

cd /home/rbrunner/bamboo\_home\_ssh/.ssh/

sudo cp plmsci1 /var/lib/jenkins/.ssh/

1. make sure the known\_hosts file gets the items from bamboos known\_host file added to the one in jenkins .ssh dir

chmod 0600 /var/lib/jenkins/.ssh/plmsci1

chown jenkins:jenkins plmsci1

1. now we have the same key bamboo uses to ssh / rysnc to the linux hosts, plmsci1, for Jenkins to use we just need to tell it to use this key and user bamboo
2. Edit the /var/lib/jenkins/.ssh/config file to add dns names of servers that Jenkins can ssh to (like how we do on Bamboo server)
3. when logged in as Jenkins user on jenkins server:

ssh -i $HOME/.ssh/plmsci1 bamboo@atl-test2corp.aff.local

# Test2corp Pipeline Project (replicating bamboo build/deploy) 11/25/2020

1. Log into Jenkins
2. New Item
3. Pipeline
4. Name: corp-test
   1. Copy from bamboo build, corp, test, configure plan
   2. General section:
      1. Description: test2corp.americanfirstfinance.com website
      2. Discard old builds:
      3. Strategy: Log Rotation
         1. Days to keep builds: (left blank)
         2. Max # of builds to keep: 3
         3. Days to keep artifacts: (left blank)
         4. Max # of builds to keep with artifacts: 3
      4. Do not allow concurrent builds
      5. Permission to Copy Artifact:
         1. Projects to allow copy artifacts: \*
   3. Build Triggers section
      1. Build when a change is pushed to Bitbucket
         1. For now Bitbucket public ips to jenkins1.aff.com 443 isnt exposed so cant used bitbucket webhooks?
         2. So temporarily using Poll SCM
            1. Poll Every 5 minutes each hour (so every 5 min)
            2. H/5 \* \* \* \*
      2. ~~Repository: americanfirstfinance/corp\_www~~
      3. ~~Branch: aws\_development~~
   4. Pipeline, Pipeline script, paste in the pipeline script (see green section above for example)
      1. **Make sure server name is updated to appropriate one, not incorrect one!**
      2. **Edit the environment variables at the tope for the appropriate:**
         1. **GIT\_REPO\_URL**
         2. **GIT\_BRANCH**
         3. **DOCKER\_IMAGE**
         4. **SERVER NAMES REFERENCED IN DEPLOY STAGE!**
   5. Click Save
   6. Click Build Project when ready
   7. Configuration, Bitbucket Endpoints
      1. Bitbucket Cloud
      2. Check: Manage Hooks
      3. Credentials:
         1. Add: Jenkins
            1. Domain Global credentials (unrestricted)
            2. Kind: ssh username with private key
            3. ID: svc\_jenkins
            4. Description: svc\_jenkins ssh private key to bitbucket cloud
            5. Username: svc\_jenkins
            6. Private key, Add, paste it in
            7. Click Add button
      4. Custom Jenkins Root URL: https://jenkins1.americanfirstfinace.com

## build/deploy Laravel app via jenkins

<https://medium.com/faun/configure-laravel-8-for-ci-cd-with-jenkins-and-github-part-1-58b9be304292>

<https://wilsonmar.github.io/jenkins2-pipeline/>

<https://stackoverflow.com/questions/33044543/what-exactly-does-archive-artifacts-step-do-in-jenkins>

"Archive the artifacts" post-build step, it copies the selected artifacts in the %JENKINS\_HOME%/jobs/MY\_JOB/builds/... on the master server.

1. Adding this into the build stage will copy all files to the appropriate projectname/builds/buildnum/archive directory

archiveArtifacts artifacts: '\*\*', onlyIfSuccessful: true

/var/lib/jenkins/jobs/corp-test/builds/63/archive

1. Had issue with rsync working when in Jenkins pipelie, selinux issue (updated Jenkins ansible playbook so it now takes care of this):

setsebool -P rsync\_client 1

setsebool -P rsync\_full\_access 1

~~setsebool -P rsync\_export\_all\_ro 1~~

Nov 27 14:58:29 pln-jenkins1 setroubleshoot[226220]: SELinux is preventing /usr/bin/rsync from execute access on the file ssh.

\*\*\*\*\* Plugin catchall\_boolean (89.3 confidence) suggests \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If you want to allow rsync to client

Then you must tell SELinux about this by enabling the 'rsync\_client' boolean.

Do

setsebool -P rsync\_client 1

1. To echo all Jenkins env variables available add this to first stage and you can see in Jenkins log:

echo sh(script: 'env|sort', returnStdout: true)

1. example of output:

BUILD\_DISPLAY\_NAME=#64

BUILD\_ID=64

BUILD\_NUMBER=64

BUILD\_TAG=jenkins-corp-test-64

BUILD\_URL=<https://jenkins1.americanfirstfinance.com/job/corp-test/64/>

EXECUTOR\_NUMBER=1

HOME=/var/lib/jenkins

HUDSON\_COOKIE=8511cbc1-db37-40ec-ad20-e64259a35c1a

HUDSON\_HOME=/var/lib/jenkins

HUDSON\_SERVER\_COOKIE=0774ad00ff1738f9

HUDSON\_URL=<https://jenkins1.americanfirstfinance.com/>

INVOCATION\_ID=a096370284f548cc95e5bfda258c0622

JENKINS\_HOME=/var/lib/jenkins

JENKINS\_NODE\_COOKIE=aed2928e-a752-4520-9c8c-317f35446284

JENKINS\_SERVER\_COOKIE=durable-a09ee7322f0e02683248af6dd6cf3658

JENKINS\_URL=<https://jenkins1.americanfirstfinance.com/>

JOB\_BASE\_NAME=corp-test

JOB\_DISPLAY\_URL=<https://jenkins1.americanfirstfinance.com/job/corp-test/display/redirect>

JOB\_NAME=corp-test

JOB\_URL=<https://jenkins1.americanfirstfinance.com/job/corp-test/>

JOURNAL\_STREAM=9:28705

LANG=en\_US.UTF-8

LOGNAME=jenkins

NODE\_LABELS=master

NODE\_NAME=master

PATH=/sbin:/usr/sbin:/bin:/usr/bin

PWD=/opt/jenkinshome1/workspace/corp-test

RUN\_ARTIFACTS\_DISPLAY\_URL=<https://jenkins1.americanfirstfinance.com/job/corp-test/64/display/redirect?page=artifacts>

RUN\_CHANGES\_DISPLAY\_URL=<https://jenkins1.americanfirstfinance.com/job/corp-test/64/display/redirect?page=changes>

RUN\_DISPLAY\_URL=<https://jenkins1.americanfirstfinance.com/job/corp-test/64/display/redirect>

RUN\_TESTS\_DISPLAY\_URL=<https://jenkins1.americanfirstfinance.com/job/corp-test/64/display/redirect?page=tests>

SHELL=/bin/bash

SHLVL=4

STAGE\_NAME=CleanWorkspace

USER=jenkins

WORKSPACE\_TMP=/var/lib/jenkins/workspace/corp-test@tmp

WORKSPACE=/var/lib/jenkins/workspace/corp-test

**Jenkins Pipeline – pass artifact url between stages**

<https://stackoverflow.com/questions/48754463/jenkins-pipeline-pass-artifact-url-between-stages>

**Configure the build stage:**

<https://docs.cloudbees.com/docs/admin-resources/latest/automating-with-jenkinsfile/config-deploy-stage>

<https://www.jenkins.io/doc/book/pipeline/jenkinsfile/>

**Clean workspaces:**

<https://stackoverflow.com/questions/58588794/what-are-the-tmp-folders-in-a-jenkins-workspace-and-how-to-clean-them-up>

## EXAMPLE PIPELINE / Jenkinsfile for test2corp.americanfirstfinance.com

1. Create Jenkinsfile (below the docker is ran just on 1st step). Had to escape the \ in the sed command \t to allow the \t to stay:

pipeline {

agent any

environment {

BITBUCKET\_CREDS = 'svc\_jenkins'

GIT\_REPO\_URL = 'git@bitbucket.org:americanfirstfinance/corp\_www.git'

GIT\_BRANCH = 'aws\_development'

DOCKER\_IMAGE = 'rbrunneraff/aff:alpine-php73-composer-gd'

}

stages {

stage('CleanWorkspace') {

steps {

cleanWs()

dir("${env.WORKSPACE}@tmp") {

deleteDir()

}

dir("${env.WORKSPACE}@script") {

deleteDir()

}

dir("${env.WORKSPACE}@script@tmp") {

deleteDir()

}

//echo sh(script: 'env|sort', returnStdout: true)

}

}

stage('Build') {

agent {

docker {

image "${env.DOCKER\_IMAGE}"

args '--net=host'

}

}

steps {

slackSend (color: '#FFFF00', message: "STARTED: Job: '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]', (${env.BUILD\_URL})")

//git branch: 'aws\_development',

git branch: "${env.GIT\_BRANCH}",

//credentialsId: 'svc\_jenkins',

//url: 'git@bitbucket.org:americanfirstfinance/corp\_www.git'

credentialsId: "${env.BITBUCKET\_CREDS}",

url: "${env.GIT\_REPO\_URL}"

sh '''

export COMPOSER\_ALLOW\_SUPERUSER=1

php /usr/local/bin/composer install

chmod -R 777 \*

'''

sh 'echo Pushing Artifacts Local Jenkins Server'

archiveArtifacts artifacts: '\*\*', onlyIfSuccessful: true

}

}

stage('Test') {

steps {

echo 'Testing...'

}

}

stage('Deploy - Dev - atl-test2corp') {

//when {

// expression {

// currentBuild.result == null || currentBuild.result == 'SUCCESS'

// }

//}

steps {

sh 'echo Current workspace: $WORKSPACE'

sh 'echo Pulling Artifacts into current workspace'

copyArtifacts projectName: "${JOB\_NAME}", selector: specific("${env.BUILD\_NUMBER}")

sh '''rsync --rsync-path="sudo rsync" -chilrvz --no-perms --no-owner --no-group --delete --exclude '.env' --exclude 'storage' --exclude 'bootstrap/cache' --exclude 'node\_modules' $WORKSPACE/. bamboo@atl-test2corp.aff.local:/opt/corp\_www'''

sh '''ssh bamboo@atl-test2corp.aff.local 'find /opt/corp\_www/\* -type d -exec sudo chmod 775 {} \\; | find /opt/corp\_www/\* -type f -exec sudo chmod 664 {} \\;' '''

sh '''ssh bamboo@atl-test2corp.aff.local 'find /opt/corp\_www/storage\* -type f -exec sudo chmod 666 {} \\; | find /opt/corp\_www/storage\* -type d -exec sudo chmod 777 {} \\; | find /opt/corp\_www/bootstrap\* -type f -exec sudo chmod 666 {} \\; | find /opt/corp\_www/bootstrap\* -type d -exec sudo chmod 777 {} \\; | sudo chown -R nginx:nginx /opt/corp\_www;' '''

//sh '''

// rsync --rsync-path="sudo rsync" -chirvz --no-perms --no-owner --no-group --delete --exclude '.env' --exclude 'storage' --exclude 'bootstrap/cache' --exclude 'rsync\_file\_changes' $WORKSPACE/. bamboo@atl-test2corp.aff.local:/opt/corp\_www | egrep -v "sending incremental file list" | sed '/^$/d' | sed '/^\*deleting/d' | sed '/^sent.\*sec$/d' | sed '/^total/d' | sed 's/^.\*[ \\t]//' | tee rsync\_file\_changes

// '''

// sh '''rsync --rsync-path="sudo rsync" rsync\_file\_changes bamboo@atl-test2corp.aff.local:/opt/corp\_www'''

// sh '''ssh bamboo@atl-test2corp.aff.local 'sudo chown nginx:nginx /opt/corp\_www/rsync\_file\_changes' '''

// sh '''ssh bamboo@atl-test2corp.aff.local 'sudo chmod 775 /opt/corp\_www/rsync\_file\_changes' '''

// sh '''ssh bamboo@atl-test2corp.aff.local 'cd /opt/corp\_www; sudo cat rsync\_file\_changes | tr "\\n" "\\0" | tee >(xargs -0 sudo chown nginx:nginx) | xargs -0 sudo chmod 664' '''

}

}

}

post {

success {

slackSend (color: '#00FF00', message: "SUCCESSFUL: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]', (${env.BUILD\_URL})")

}

failure {

slackSend (color: '#FF0000', message: "FAILED: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]' Stage: (${env.STAGE\_NAME}) (${env.BUILD\_URL})")

}

}

}

## ~~Matrix-based security:~~

1. ~~Add user or group:~~ 
   1. ~~Developers~~
   2. ~~QA~~
   3. ~~DevOps~~