My Jenkins Project

Jenkins Installation follow the configuration in the link below.

<https://github.com/devopsjourney1/jenkins-101> - Old

<https://phoenixnap.com/kb/install-jenkins-on-mac> - Latest

<https://github.com/jenkinsci/docker/blob/master/README.md>

Jenkins initial setup is required. An admin user has been created and a password generated.

Please use the following password to proceed to installation:

6a98e973a1f44e8b91da78ea7b324e32

This may also be found at: /var/jenkins\_home/secrets/initialAdminPassword

Graphical user interface, text, application, Teams

Description automatically generated

112b8303baa44a799c2316cfbacecf9f21

Telegram Bot

Use this token to access the HTTP API:

My\_Jenkins\_CICD\_bot ( **5796029305:AAFAoFwCS9\_Y55ax5hci77JKXTahl-z8NCg** )

Chat\_ID=1686082402

To integrate to github follow the instruction below

<https://docs.github.com/en/developers/webhooks-and-events/webhooks/creating-webhooks>

<https://dashboard.ngrok.com/get-started/setup>

% brew reinstall --cask ngrok

Paste your token

% ngrok config add-authtoken 2F5O2YHgsg9ajY14FZ8RNpBClPP\_3CdA2wss3e6FemYU2RbtL

Then run this command

% ngrok http 8080

Graphical user interface, text, application, chat or text message

Description automatically generated

Text

Description automatically generated

Server Reboot

Aborted and Approved testing



pipeline {

agent any

environment {

uat\_bis = 'UAT-bis'

uat\_cde = 'UAT-cde'

uat\_ebiller = 'UAT-ebiller'

uat\_fileserver = 'UAT-fileserver'

uat\_gateway = 'UAT-gateway'

uat\_ncde = 'UAT-ncde'

uat\_PaygateHVM = 'UAT-PaygateHVM'

uat\_ptiapps = 'UAT-ptiapps'

uat\_ptipaygate = 'UAT-ptipaygate'

uat\_recon = 'UAT-recon'

uat\_webterminal = 'UAT-webterminal'

// Instance IDs to be rebooted

// id\_uat\_bis = 'i-07b575a118ff5b1a6'

// id\_uat\_cde = 'i-073b53d62cae3aefe'

// id\_uat\_ebiller = 'i-0374825351e79bba3'

// id\_uat\_fileserver = 'i-05803b05bf4fc44d8'

// id\_uat\_gateway = 'i-0a94792077a36c8cd'

// id\_uat\_ncde = 'i-008ae1dfaecfdf662'

// id\_uat\_PaygateHVM = 'i-0b5e0a696afd25156'

// id\_uat\_ptiapps = 'i-0b0dfdd46256b341e'

// id\_uat\_ptipaygate = 'i-08226d3452e70e8ac'

// id\_uat\_recon = 'i-04e5f8656c0c834ee'

// id\_uat\_webterminal = 'i-0a30b880bfc8e1126'

}

stages {

stage('Push Notification: Start') {

steps {

echo '=========== Notification: Reboot Started ============'

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<strong>Notice!, UAT Servers Reboot started! \n\n Affected Servers: </strong> <i> \n 👉 ${uat\_bis} \n 👉 ${uat\_cde} \n 👉 ${uat\_ebiller} \n 👉 ${uat\_fileserver} \n 👉 ${uat\_gateway} \n 👉 ${uat\_ncde} \n 👉 ${uat\_PaygateHVM} \n 👉 ${uat\_ptiapps} \n 👉 ${uat\_ptipaygate} \n 👉 ${uat\_recon} \n 👉 ${uat\_webterminal}</i>"'

}

}

}

stage('Reboot Approval') {

steps {

echo '===========Push Notification: Approval==========='

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>Pause! Automated Reboot for UAT Servers for approval.</b>"'

script {

def proceed = true

try {

timeout(time: 3600, unit: 'SECONDS') {

input('Do you want to proceed for production deployment?')

}

sh 'echo "Deployment to production will resume"'

} catch (err) {

proceed = false

currentBuild.result = 'ABORTED'

}

if (proceed) {

echo '=========== PROCEED ==========='

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>Servers Reboot for UAT all server, Approved!</b>"'

} else {

echo '=========== DONT PROCEED ==========='

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>Servers Reboot Aborted! Approval is needed.</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

}

}

}

}

stage('Server Reboot') {

steps {

echo '=========== SERVER REBOOT ============'

script {

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

try {

echo '=========== REBOOT: ${uat\_bis} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_bis}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_bis} reboot completed! Server is running.</b>"'

} catch (Exception e) {

echo '=========== REBOOT Failure ==========='

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_bis} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_cde} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_cde}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_cde} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_cde} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_ebiller} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_ebiller}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_ebiller} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_ebiller} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_fileserver} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_fileserver}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_fileserver} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_fileserver} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_gateway} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_gateway}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_gateway} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_gateway} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_ncde} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_ncde}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_ncde} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_ncde} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_PaygateHVM} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_PaygateHVM}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_PaygateHVM} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_PaygateHVM} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_ptiapps} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_ptiapps}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_ptiapps} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_ptiapps} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_ptipaygate} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_ptipaygate}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_ptipaygate} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_ptipaygate} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_recon} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_recon}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_recon} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_recon} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

try {

echo '=========== REBOOT: ${uat\_webterminal} ============'

// sh 'aws ec2 reboot-instances --instance-ids ${id\_uat\_webterminal}'

// sleep(30)

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_webterminal} reboot completed! Server is running.</b>"'

} catch (Exception e) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>${uat\_webterminal} server reboot, Failed!</b>"'

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

}

}

}

}

stage('Push Notification: End') {

steps {

echo '=========== Notification: Reboot Success ============'

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \

-d chat\_id=${CHAT\_ID} \

-d parse\_mode="HTML" \

-d text="<b>Congratulations! UAT Servers Reboot on Paynamics Account ID are successfully Completed!</b>"'

}

}

}

}

}

Terraform Test (AMI Backup)

Aborted and Approved testing



pipeline{

agent any

//{

// label 'jenkins-paynamics'

//}

//tools {

// terraform 'terraform-11'

//}

//environment {

// service = "AMI Backup (Paynamics | Oregon)"

//}

stages{

stage ('Push Notification: Start') {

steps{

echo '=========== Notification: Build Started ============'

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Hello, Terraform build started on ${service}.</b>"'

}

}

}

stage ('Git Checkout') {

steps{

echo '=========== GIT CHECKOUT ============'

script {

try {

echo '=========== GIT CLONE ============'

// git credentialsId: 'git-jenkins-key', url: 'git@gitlab.paynamics.net:terraform/aws-ec2-ami-paynamics-oregon-02.git'

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Gitlab checkout for ${service}, Successuful!</b>"'

}

} catch (Exception e) {

echo '=========== GIT Clone Failure ==========='

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Gitlab checkout for ${service}, Failed! Please Check. </b>"'

}

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

}

}

}

stage ('Terraform Init') {

steps{

script {

try {

echo '=========== Terraform Init ============'

// sh 'terraform init'

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Terraform Initialization for ${service}, Successful!</b>"'

}

} catch (Exception e) {

echo '=========== Terraform Init Failure ==========='

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Terraform Init for ${service} is Failed!</b>"'

}

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

}

}

}

stage ('Terraform Plan') {

steps{

script {

try {

echo '=========== Terraform Plan ============'

// sh 'terraform plan'

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Terraform Plan for ${service}, Successful!</b>"'

}

} catch (Exception e) {

echo '=========== Terraform Plan Failure ==========='

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Terraform Plan for ${service}, Failed!</b>"'

}

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

}

}

}

stage ('Push Notification: Under Review') {

steps {

echo '===========Push Notification: Approval==========='

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Infrastructure changes for ${service} is under review! Please wait until the reviewer approve the changes</b>"'

}

}

}

stage ('Terraform Approval') {

steps {

script {

def proceed = true

try {

timeout(time: 3600, unit: 'SECONDS') {

input('Do you want to proceed for production deployment?')

}

sh 'echo "Deployment to production will resume"'

} catch (err) {

proceed = false

currentBuild.result = 'ABORTED'

}

if(proceed) {

echo '=========== PROCEED ==========='

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Infrastructure changes for ${service}, Approved!</b>"'

}

} else {

echo '=========== DONT PROCEED ==========='

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Infrastructure changes Aborted! Approval is needed for ${service}.</b>"'

}

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

}

}

}

stage ('Terraform Apply') {

steps{

script {

try {

echo '=========== Terraform Apply ============'

// sh 'terraform apply --auto-approve'

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Terraform Apply for ${service}, Successful!</b>"'

}

} catch (Exception e) {

echo '=========== Terraform Apply Failure ==========='

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Terraform Apply for ${service}, Failed!</b>"'

}

currentBuild.result = 'ABORTED'

error('Stopping early…')

}

}

}

}

stage ('Push Notification: End') {

steps{

echo '=========== Notification: Build Success ============'

withCredentials([string(credentialsId: 'telegramToken', variable: 'TOKEN'),

string(credentialsId: 'telegramChatID', variable: 'CHAT\_ID')]) {

sh 'curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat\_id=${CHAT\_ID} -d parse\_mode="HTML" -d text="<b>Congratulations! Infrastructure changes fo ${service} is successfully Completed!</b>"'

}

}

}

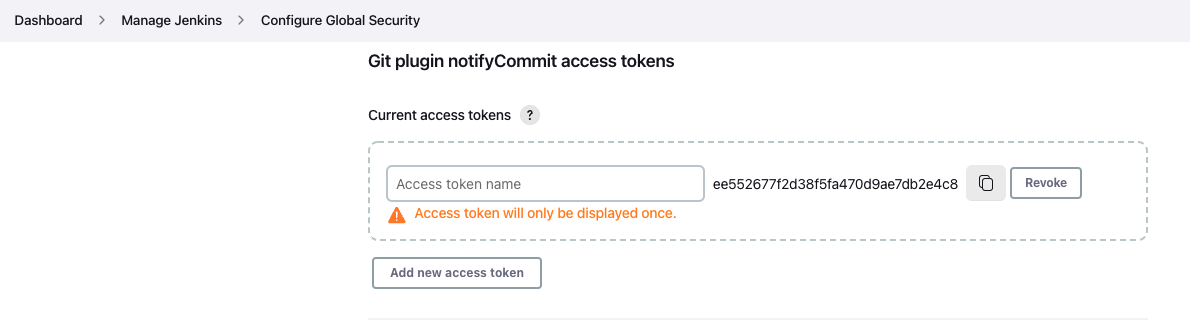
}

}

How to generate git token

Goto [Dashboard](https://5e68-35-163-200-179.ngrok.io/) -> [Manage Jenkins](https://5e68-35-163-200-179.ngrok.io/manage/) -> [Configure Global Security](https://5e68-35-163-200-179.ngrok.io/manage/configureSecurity/)

ee552677f2d38f5fa470d9ae7db2e4c8



Create GPG key

https://gist.github.com/phortuin/cf24b1cca3258720c71ad42977e1ba57