**USE CASES – DevOps**

**Table of Contents**

[Use Case 1: Create 5 Droplets 5](#_Toc13229734)

[Use Case 2: Change Password for Job Droplet 7](#_Toc13229735)

[Use Case 3: Change Password for Master Droplet 8](#_Toc13229736)

[Use Case 4: Change Password for Node Droplet for Onboarding 9](#_Toc13229737)

[Use Case 5: Change Password for Node Droplet for Platform 10](#_Toc13229738)

[Use Case 6: Change Password for Nexus Droplet 11](#_Toc13229739)

[Use Case 7: Create Ansible Script – Install Jenkins, JQuery & Docker 12](#_Toc13229740)

[Use Case 8: Create Ansible Script – Install Nexus 15](#_Toc13229741)

[Use Case 9: Create Ansible Script – Install Kubernetes 19](#_Toc13229742)

[Use Case 10: Create Ansible Script – To copy SSL Certification from GIT to Node Droplet 24](#_Toc13229743)

[Use Case 11: Create Ansible Script – To configure Docker 26](#_Toc13229744)

[Use Case 12: Create Ansible Script – To configure New nodes 28](#_Toc13229745)

[Use Case 13: Create Ansible Script – To deploy Onboarding 32](#_Toc13229746)

[Use Case 14: Create Ansible Script – To deploy Platform 37](#_Toc13229747)

[Use Case 15: Create a Repository in GitHub 42](#_Toc13229748)

[Use Case 16: Push Ansible Scripts to the Repository 43](#_Toc13229749)

[Use Case 17: Install Ansible – Job Droplet 44](#_Toc13229750)

[Use Case 18: Install Git Client – Job Droplet 45](#_Toc13229751)

[Use Case 19: Run Jenkin Installation Ansible Script – Job Droplet 46](#_Toc13229752)

[Ue Case 20: Create User in Jenkins 47](#_Toc13229753)

[Use Case 21: Create Credentials in Jenkins for GitHub 50](#_Toc13229754)

[Use Case 22: Create Jenkins Job – To run Onboarding Front End Script 51](#_Toc13229755)

[Use Case 23: Create Jenkins Job – To run Onboarding Back End Script 52](#_Toc13229756)

[Use Case 24: Create Jenkins Job – To run Onboarding Key Cloak 53](#_Toc13229757)

[Use Case 25: Create Jenkins Job – To run Onboarding Billing 54](#_Toc13229758)

[Use Case 26: Create Jenkins Job – To run Platform Front End 55](#_Toc13229759)

[Use Case 27: Create Jenkins Job – To run Platform Back End 56](#_Toc13229760)

[Use Case 28: Create Jenkins Job – To run Platform Key Cloak 57](#_Toc13229761)

[Use Case 29: Create Jenkins Job – To run Platform Elastic Search 58](#_Toc13229762)

[Use Case 30: Create Jenkins Job – To run Platform Image Service 59](#_Toc13229763)

[Use Case 31: Create Jenkins Job – To run Nexus Installation Ansible Script 61](#_Toc13229764)

[Use Case 32: Create Jenkins Job – To run SSL certification copy Ansible script 62](#_Toc13229765)

[Use Case 33: Create Jenkins Job – To run Ansible scripts of Kubernetes and Docker Installation with Configuration & Nexus Configuration 64](#_Toc13229766)

[Use Case 34: Create Jenkins Job – To run Ansible scripts of New Node Configuration 66](#_Toc13229767)

[Use Case 35: Create Jenkins Job – To Mount DNS to IP 68](#_Toc13229768)

[Use Case 36: Create Jenkins Job – To run Ansible script of Onboarding Deployment 69](#_Toc13229769)

[Use Case 37: Create Jenkins Job – To run Ansible script of Platform Deployment 73](#_Toc13229770)

[Use Case 38: Run Nexus Installation Jenkins Job 78](#_Toc13229771)

[Use Case 39: Configure Nexus – Nexus Droplet 79](#_Toc13229772)

[Use Case 40: Create Credentials in Jenkins for Nexus 81](#_Toc13229773)

[Use Case 41: Configure Docker – Job Droplet 82](#_Toc13229774)

[Use Case 42: Update Nexus Credentials – Onboarding Frontend 84](#_Toc13229775)

[Use Case 43: Update Nexus Credentials – Onboarding Backend 85](#_Toc13229776)

[Use Case 44: Update Nexus Credentials – Onboarding Keycloak 86](#_Toc13229777)

[Use Case 45: Update Nexus Credentials – Onboarding Billing 87](#_Toc13229778)

[Use Case 46: Update Nexus Credentials – Platform Frontend 88](#_Toc13229779)

[Use Case 47: Update Nexus Credentials – Platform Backend 89](#_Toc13229780)

[Use Case 48: Update Nexus Credentials – Platform Keycloak 90](#_Toc13229781)

[Use Case 49: Update Nexus Credentials – Platform Elasticsearch Backend 91](#_Toc13229782)

[Use Case 50: Update Nexus Credentials – Platform Image Service 92](#_Toc13229783)

[Use Case 51: Run Onboarding Front End Jenkins Job 93](#_Toc13229784)

[Use Case 52: Run Onboarding Back End Jenkins Job 94](#_Toc13229785)

[Use Case 53: Run Onboarding Key Cloak Jenkins Job 95](#_Toc13229786)

[Use Case 54: Run Onboarding Billing Jenkins Job 96](#_Toc13229787)

[Use Case 55: Run Platform Front End Jenkins Job 97](#_Toc13229788)

[Use Case 56: Run Platform Back End Jenkins Job 98](#_Toc13229789)

[Use Case 57: Run Platform Key Cloak Jenkins Job 99](#_Toc13229790)

[Use Case 58: Run Platform Elastic Search Jenkins Job 100](#_Toc13229791)

[Use Case 59: Run Platform Image Service Jenkins Job 101](#_Toc13229792)

[Use Case 60: Update Nexus Credentials - Kubernetes 102](#_Toc13229793)

[Use Case 61: Run Kubernetes and Docker Installation with Configuration & Nexus Configuration Jenkins Job 103](#_Toc13229794)

[Use Case 62: Update Nexus Credentials – New Node Kubernetes 104](#_Toc13229795)

[Use Case 63: Run Kubernetes and Docker Installation with Configuration & Nexus Configuration Jenkins Job for New Node. 105](#_Toc13229796)

[Use Case 64: Run SSL certification copy Jenkins Job - Onboarding 106](#_Toc13229797)

[Use Case 65: Run SSL certification copy Jenkins Job - Platform 107](#_Toc13229798)

[Use Case 66: Run DNS mount Jenkins Job - Onboarding 108](#_Toc13229799)

[Use Case 67: Run DNS mount Jenkins Job - Platform 109](#_Toc13229800)

[Use Case 68: Update Nexus Credentials with version updates – Onboarding Deployment 110](#_Toc13229801)

[Use Case 69: Update Nexus Credentials with version updates – Platform Deployment 113](#_Toc13229802)

[Use Case 70: Run Onboarding Deploy Jenkins Job 116](#_Toc13229803)

## Use Case 1: Create 5 Droplets

**Actor:** DevOps Member

**Description:** To initiate the process of DevOps, Actor needs to create 5 Droplets. Using these Droplets, actor will automate the necessary tools to make the deployment seamless. The necessity of the 5 Droplets are as follows:

* Droplet 1 – Job Droplet - To run the deployment process and save the versions of the code
* Droplet 2 – Master Droplet – To pull the code from Job Droplet and push it to Node Droplet
* Droplet 3 – Node Droplet – To run the deployed code of Onboarding in the website for use
* Droplet 4 – Node Droplet – To run the deployed code of platform in the website for use
* Droplet 5 – Nexus Droplet – To store the images involved in the website

**Pre – Condition:** Actor needs to have a Digital Ocean Account and logins to the account to create the droplets

**Normal Workflow:**

Actor clicks “Create Droplet”

Actor selects the version of the Droplet

***CentOS version 7.6***

Actor selects an option under “Choose a Size”

***$20 per month***

Actor selects an option under “Choose a datacenter region”

***London***

Actor selects Additional options, if needed

Actor fills “5 Droplets” under “How many Droplets?”

Actor fills a name under “Choose a Hostname”

Actor clicks “Create”

**Alternate Workflow:**

**Post – Condition:** 5 Droplets will be created with their respective IP Address. Actor receives an email from Digital ocean about the newly created Droplets with their IP address, Username and Password.

## Use Case 2: Change Password for Job Droplet

**Actor:** DevOps Member

**Description:** For security measures, Actor needs to change the password for the Job Droplet they created.

**Pre – Condition:** Actor receives an email from Digital ocean about the newly created Droplets with their IP address, Username and Password.

**Normal Workflow:**

Actor opens the Terminal

***ssh root@<Job Droplet IP>***

Actor connects the droplet with the username received in Email using SSH authentication

Actor confirms to continue the connection by typing “Yes”

Actor types the password received in the email

Actor asks to change the password for the Droplet

Actor types the current password

Actor types new password

Actor re-types new password

**Alternate Workflow:**

**Post – Condition:** Actor gets access to the Job Droplet Server

## Use Case 3: Change Password for Master Droplet

**Actor:** DevOps Member

**Description:** For security measures, Actor needs to change the password for the Master Droplet they created.

**Pre – Condition:** Actor receives an email from Digital ocean about the newly created Droplets with their IP address, Username and Password.

**Normal Workflow:**

Actor opens the Terminal

***ssh root@<Master Droplet IP>***

Actor connects the droplet with the username received in Email using SSH authentication

Actor confirms to continue the connection by typing “Yes”

Actor types the password received in the email

Actor asks to change the password for the Droplet

Actor types the current password

Actor types new password

Actor re-types new password

**Alternate Workflow:**

**Post – Condition:** Actor gets access to the Master Droplet Server

## Use Case 4: Change Password for Node Droplet for Onboarding

**Actor:** DevOps Member

**Description:** For security measures, Actor needs to change the password for the Node Droplet they created.

**Pre – Condition:** Actor receives an email from Digital ocean about the newly created Droplets with their IP address, Username and Password.

**Normal Workflow:**

Actor opens the Terminal

***ssh root@<Onboarding Node Droplet IP - Onboarding>***

Actor connects the droplet with the username received in Email using SSH authentication

Actor confirms to continue the connection by typing “Yes”

Actor types the password received in the email

Actor asks to change the password for the Droplet

Actor types the current password

Actor types new password

Actor re-types new password

**Alternate Workflow:**

**Post – Condition:** Actor gets access to the Node Droplet Server for onboarding

## Use Case 5: Change Password for Node Droplet for Platform

**Actor:** DevOps Member

**Description:** For security measures, Actor needs to change the password for the Node Droplet they created.

**Pre – Condition:** Actor receives an email from Digital ocean about the newly created Droplets with their IP address, Username and Password.

**Normal Workflow:**

1. Actor opens the Terminal

***ssh root@<Onboarding Node Droplet IP - Platform>***

1. Actor connects the droplet with the username received in Email using SSH authentication
2. Actor confirms to continue the connection by typing “Yes”
3. Actor types the password received in the email
4. Actor asks to change the password for the Droplet
5. Actor types the current password
6. Actor types new password
7. Actor re-types new password

**Alternate Workflow:**

**Post – Condition:** Actor gets access to the Node Droplet Server for platform

## Use Case 6: Change Password for Nexus Droplet

**Actor:** DevOps Member

**Description:** For security measures, Actor needs to change the password for the Nexus Droplet they created.

**Pre – Condition:** Actor receives an email from Digital ocean about the newly created Droplets with their IP address, Username and Password.

**Normal Workflow:**

1. Actor opens the Terminal

***ssh root@<Nexus Droplet IP>***

1. Actor connects the droplet with the username received in Email using SSH authentication
2. Actor confirms to continue the connection by typing “Yes”
3. Actor types the password received in the email
4. Actor asks to change the password for the Droplet
5. Actor types the current password
6. Actor types new password
7. Actor re-types new password

**Alternate Workflow:**

**Post – Condition:** Actor gets access to the Nexus Droplet Server

## Use Case 7: Create Ansible Script – Install Jenkins, JQuery & Docker

**Actor:** DevOps Member

**Description:** Actor creates Ansible script with the instructions to install Jenkins, JQuery and Docker inside the Job Droplet

**Pre – Condition:** Actor can use any software or notepad to write the script

**Normal Workflow:**

Actor writes the script to install Jenkins, JQuery & Docker inside the Job Droplet. Steps involved

***---***

***- hosts: jenkins***

***tasks:***

***- name: Installing java8***

***command: yum install -y java-1.8.0-openjdk-devel***

***sudo: true***

***- name: Ensure Jenkins Repository is Installed***

***yum\_repository:***

***name: jenkins***

***state: present***

***description: Official Jenkins Yum Repo***

***baseurl: http://pkg.jenkins.io/redhat***

***gpgkey: https://jenkins-ci.org/redhat/jenkins-ci.org.key***

***gpgcheck: yes***

***enabled: yes***

***- name: Ensure Jenkins is Installed***

***yum :***

***name: jenkins***

***update\_cache: yes***

***state: present***

***- name: Enable and Start the Jenkins Service***

***service:***

***name: jenkins***

***enabled: yes***

***state: started***

***sudo: true***

***- name: Installing Wget for JQ***

***shell: yum install -y Wget***

***- name: Installing the JQ***

***shell: wget http://download-ib01.fedoraproject.org/pub/epel/7/x86\_64/ && yum install -y jq***

***- name: "Installing Docker Prerequisite packages"***

***yum:***

***name: "{{ item }}"***

***state: latest***

***with\_items:***

***- yum-utils***

***- device-mapper-persistent-data***

***- lvm2***

***- name: "Configuring docker-ce repo"***

***get\_url:***

***url: https://download.docker.com/linux/centos/docker-ce.repo***

***dest: /etc/yum.repos.d/docker-ce.repo***

***mode: 0644***

***- name: "Installing Docker latest version"***

***yum:***

***name: docker-ce***

***state: present***

***- name: "Starting and Enabling Docker service"***

***service:***

***name: docker***

***state: started***

***enabled: yes***

Actor saves the script file

**Alternate Workflow:** None

**Post – Condition:** Ansible script is ready to install Jenkins, JQuery & Docker

## Use Case 8: Create Ansible Script – Install Nexus

**Actor:** DevOps Member

**Description:** Actor creates Ansible script with the instructions to install Nexus inside the Nexus Droplet

**Pre – Condition:** Actor can use any software or notepad to write the script

**Normal Workflow:**

Actor writes the script to install Nexus inside the Nexus Droplet. Steps involved to install Nexus

***---***

***- hosts: nexus***

***sudo: true***

***vars:***

***nexus\_version: latest***

***nexus\_download\_url: https://download.sonatype.com/nexus/3/{{nexus\_version}}-unix.tar.gz***

***#nexus\_download\_url: http://www.sonatype.org/downloads/nexus-{{nexus\_version}}-bundle.tar.gz***

***nexus\_download\_dest: /tmp/nexus-{{nexus\_version}}-unix.tar.gz***

***nexus\_extracted\_dir: /tmp/nexus\_extracted\_dir***

***nexus\_dir: /opt/nexus***

***nexus\_root: /opt***

***tasks:***

***#- name: update yum repo***

***#yum:***

***#update\_cache: true***

***#sudo: true***

***- name: install wget***

***yum:***

***name: wget***

***state: present***

***update\_cache: true***

***- name: Installing java 8***

***command: yum install -y java-1.8.0-openjdk-devel***

***- name: create "nexus" group***

***group: name=nexus***

***sudo: true***

***- name: create "nexus" user***

***user: name=nexus group=nexus***

***sudo: true***

***- name: download nexus***

***get\_url: url={{nexus\_download\_url}} dest={{nexus\_download\_dest}}***

***register: nexus\_download***

***- name: create {{nexus\_extracted\_dir}} directory***

***file: path={{nexus\_extracted\_dir}} state=directory***

***when: nexus\_download.changed***

***sudo: true***

***- name: extract nexus to {{nexus\_extracted\_dir}}***

***command: tar xzf {{nexus\_download\_dest}} -C {{nexus\_extracted\_dir}} --strip-components=1***

***when: nexus\_download.changed***

***sudo: true***

***- name: move nexus to {{nexus\_dir}} directory***

***command: cp -a {{nexus\_extracted\_dir}}/. {{nexus\_dir}}***

***when: nexus\_download.changed***

***sudo: true***

***- name: remove {{nexus\_extracted\_dir}} directory***

***command: rm -rf {{nexus\_extracted\_dir}}***

***when: nexus\_download.changed***

***sudo: true***

***- name: make {{nexus\_root}} directory property of "nexus" user/group***

***file: path={{nexus\_root}} group=nexus owner=nexus recurse=true***

***sudo: true***

***- name: make nexus runned by "nexus" user***

***lineinfile: dest={{nexus\_dir}}/bin/nexus.rc regexp="#run\_as\_user=" line="run\_as\_user=nexus" backrefs=true***

***sudo: true***

***- name: set NEXUS\_HOME***

***lineinfile: dest={{nexus\_dir}}/bin/nexus regexp="^NEXUS\_HOME" line="NEXUS\_HOME={{nexus\_dir}}" backrefs=true***

***sudo: true***

***- name: create nexus piddir***

***file: path=/var/run/nexus state=directory group=nexus owner=nexus***

***sudo: true***

***- name: set nexus pidir***

***lineinfile: dest={{nexus\_dir}}/bin/nexus regexp="^#PIDDIR=" line="PIDDIR=/var/run/nexus" backrefs=true***

***sudo: true***

***- name: create symbolic links to /etc/init.d/nexus***

***file: src={{nexus\_dir}}/bin/nexus dest=/etc/init.d/nexus state=link***

***sudo: true***

***- name: configure runlevel links for nexus***

***command: chkconfig --add nexus***

***command: chkconfig --levels 345 nexus on***

***sudo: true***

***- name: start nexus***

***service: name=nexus state=started pattern={{nexus\_dir}}***

***sudo: true***

Actor saves the script file

**Alternate Workflow:** None

**Post – Condition:** Ansible script is ready to install Nexus

## Use Case 9: Create Ansible Script – Install Kubernetes

**Actor:** DevOps Member

**Description:** Actor creates Ansible script with the instructions to install Kubernetes inside the Master and Node Droplet. The script for installing Docker gets included along with this.

**Pre – Condition:** Actor can use any software or notepad to write the script

**Normal Workflow:**

Actor writes the script to install Kubernetes inside the Master and Node Droplet. Steps involved to install Kubernetes

***---***

***- hosts: all***

***become: yes***

***tasks:***

***- name: "Installing Docker Prerequisite packages"***

***yum:***

***name: "{{ item }}"***

***state: latest***

***with\_items:***

***- yum-utils***

***- device-mapper-persistent-data***

***- lvm2***

***- name: "Configuring docker-ce repo"***

***get\_url:***

***url: https://download.docker.com/linux/centos/docker-ce.repo***

***dest: /etc/yum.repos.d/docker-ce.repo***

***mode: 0644***

***- name: " Installing Docker latest version"***

***yum:***

***name: docker-ce***

***state: present***

***- name: " Starting and Enabling Docker service"***

***service:***

***name: docker***

***state: started***

***enabled: yes***

***- name: disable SELinux***

***command: setenforce 0***

***- name: disable SELinux on reboot***

***selinux:***

***state: disabled***

***- name: ensure net.bridge.bridge-nf-call-ip6tables is set to 1***

***sysctl:***

***name: net.bridge.bridge-nf-call-ip6tables***

***value: 1***

***state: present***

***- name: ensure net.bridge.bridge-nf-call-iptables is set to 1***

***sysctl:***

***name: net.bridge.bridge-nf-call-iptables***

***value: 1***

***state: present***

***- name: add Kubernetes' YUM repository***

***yum\_repository:***

***name: Kubernetes***

***description: Kubernetes YUM repository***

***baseurl: https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64***

***gpgkey: https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg***

***gpgcheck: yes***

***- name: install kubelet***

***yum:***

***name: kubelet***

***state: present***

***update\_cache: true***

***- name: install kubeadm***

***yum:***

***name: kubeadm***

***state: present***

***- name: start kubelet***

***service:***

***name: kubelet***

***enabled: yes***

***state: started***

***- hosts: master***

***become: yes***

***tasks:***

***- name: install kubectl***

***yum:***

***name: kubectl***

***state: present***

***- name: initialize the cluster***

***shell: kubeadm init --pod-network-cidr=10.244.0.0/16***

***#shell: kubeadm init --pod-network-cidr=10.244.0.0/16 >> cluster\_initialized.txt***

***#args:***

***# chdir: $HOME***

***# creates: cluster\_initialized.txt***

***- name: create .kube directory***

***become: yes***

***become\_user: centos***

***file:***

***path: $HOME/.kube***

***state: directory***

***mode: 0755***

***- name: copy admin.conf to user's kube config***

***copy:***

***src: /etc/kubernetes/admin.conf***

***dest: /home/centos/.kube/config***

***remote\_src: yes***

***owner: centos***

***- name: install Pod network***

***become: yes***

***become\_user: centos***

***#shell: kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/v0.9.1/Documentation/kube-flannel.yml >> pod\_network\_setup.txt #This Flanel has an issue which makes the coredns to stop creating***

***shell: kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/bc79dd1505b0c8681ece4de4c0d86c5cd2643275/Documentation/kube-flannel.yml >> pod\_network\_setup.txt***

***#shell: https://raw.githubusercontent.com/coreos/flannel/bc79dd1505b0c8681ece4de4c0d86c5cd2643275/Documentation/kube-flannel.yml >> pod\_network\_setup.txt***

***args:***

***chdir: $HOME***

***creates: pod\_network\_setup.txt***

***- hosts: master***

***become: yes***

***gather\_facts: false***

***tasks:***

***- name: get join command***

***shell: kubeadm token create --print-join-command***

***register: join\_command\_raw***

***- name: set join command***

***set\_fact:***

***join\_command: "{{ join\_command\_raw.stdout\_lines[0] }}"***

***- hosts: workers***

***become: yes***

***tasks:***

***- name: join cluster***

***shell: "{{ hostvars['master'].join\_command }} >> node\_joined.txt"***

***args:***

***chdir: $HOME***

***creates: node\_joined.txt***

***- name: Setting VM max size***

***command: sysctl -w vm.max\_map\_count=262144***

Actor saves the script file

**Alternate Workflow:** None

**Post – Condition:** Ansible script is ready to install Kubernetes

## Use Case 10: Create Ansible Script – To copy SSL Certification from GIT to Node Droplet

**Actor:** DevOps Member

**Description:** Actor creates Ansible script with the instructions to copy the SSL Certificate from GIT to Node Droplet

**Pre – Condition:** Actor can use any software or notepad to write the script

**Normal Workflow:**

1. Actor writes the script to copy SSL Certification. Steps involved to copy SSL Certification

***---***

***- hosts: workers***

***become\_user: centos***

***vars:***

***folder\_name: test***

***tasks:***

***- name: Install Git in node***

***command: yum install git -y***

***- name: Get updated files from git repository***

***git:***

***repo: "https://sysdevopsCC:QiNEToGafiPejatr5@github.com/ComplianceCompedium/CCS-Onboarding-DevOps.git"***

***dest: "/tmp/wildcardssl/"***

***accept\_hostkey: yes***

***clone: yes***

***version: development***

***- name: Creates directory***

***file:***

***path: /etc/letsencrypt/live***

***state: directory***

***- name: copy cccseu.com folder***

***command: cp -r /tmp/wildcardssl/Centos/wildcard\_ssl/{{ folder\_name }} /etc/letsencrypt/live/***

1. Actor saves the script file

**Alternate Workflow:** None

**Post – Condition:** Ansible script is ready to copy the SSL Certificate from GIT to Node Droplet

## Use Case 11: Create Ansible Script – To configure Docker

**Actor:** DevOps Member

**Description:** Actor creates Ansible script with the instructions to configure Docker.

**Pre – Condition:** Actor can use any software or notepad to write the script

**Normal Workflow:**

1. Actor writes the script to configure Docker. Steps involved to configure Docker

***---***

***- hosts: all***

***become: yes***

***vars:***

***nexus\_user: abc #dummy user***

***nexus\_pwd: abc2323 #dummy pwd***

***nexus\_ip: 152.343.545.6 #dummy ip***

***nexus\_docker\_port: 8110 #dummy port***

***tasks:***

***- name: Changing the docker /var/run/docker.sock permission to docker group***

***shell: chown root:docker /var/run/docker.sock***

***- name: Adding the centos user to the group***

***shell: usermod -a -G docker centos***

***- name: Copy the docker.json***

***copy:***

***src: ./daemon.json***

***dest: /etc/docker/***

***become: yes***

***- name: restart docker***

***service:***

***name: docker***

***state: restarted***

***- name: Login to the nexus Artifactory***

***command: docker login -u {{nexus\_user}} -p {{nexus\_pwd}} {{nexus\_ip}}:{{nexus\_docker\_port}}***

1. Actor saves the script file

**Alternate Workflow:** None

**Post – Condition:** Ansible script is ready to configure Docker

## Use Case 12: Create Ansible Script – To configure New nodes

**Actor:** DevOps Member

**Description:** Actor creates Ansible script with the instructions to configure new Nodes

**Pre – Condition:** Actor can use any software or notepad to write the script

**Normal Workflow:**

1. Actor writes the script to configure New Nodes. Steps involved to configure New Node

***---***

***- hosts: workers***

***become: yes***

***tasks:***

***- name: "Installing Docker Prerequisite packages"***

***yum:***

***name: "{{ item }}"***

***state: latest***

***with\_items:***

***- yum-utils***

***- device-mapper-persistent-data***

***- lvm2***

***- name: "Configuring docker-ce repo"***

***get\_url:***

***url: https://download.docker.com/linux/centos/docker-ce.repo***

***dest: /etc/yum.repos.d/docker-ce.repo***

***mode: 0644***

***- name: " Installing Docker latest version"***

***yum:***

***name: docker-ce***

***state: present***

***- name: " Starting and Enabling Docker service"***

***service:***

***name: docker***

***state: started***

***enabled: yes***

***- name: disable SELinux***

***command: setenforce 0***

***- name: disable SELinux on reboot***

***selinux:***

***state: disabled***

***- name: ensure net.bridge.bridge-nf-call-ip6tables is set to 1***

***sysctl:***

***name: net.bridge.bridge-nf-call-ip6tables***

***value: 1***

***state: present***

***- name: ensure net.bridge.bridge-nf-call-iptables is set to 1***

***sysctl:***

***name: net.bridge.bridge-nf-call-iptables***

***value: 1***

***state: present***

***- name: add Kubernetes' YUM repository***

***yum\_repository:***

***name: Kubernetes***

***description: Kubernetes YUM repository***

***baseurl: https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64***

***gpgkey: https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg***

***gpgcheck: yes***

***- name: install kubelet***

***yum:***

***name: kubelet***

***state: present***

***update\_cache: true***

***- name: install kubeadm***

***yum:***

***name: kubeadm***

***state: present***

***- name: start kubelet***

***service:***

***name: kubelet***

***enabled: yes***

***state: started***

***- hosts: master***

***become: yes***

***gather\_facts: false***

***tasks:***

***- name: get join command***

***shell: kubeadm token create --print-join-command***

***register: join\_command\_raw***

***- name: set join command***

***set\_fact:***

***join\_command: "{{ join\_command\_raw.stdout\_lines[0] }}"***

***- hosts: workers***

***become: yes***

***tasks:***

***- name: join cluster***

***shell: "{{ hostvars['master'].join\_command }} >> node\_joined.txt"***

***args:***

***chdir: $HOME***

***creates: node\_joined.txt***

***- name: Setting VM max size***

***command: sysctl -w vm.max\_map\_count=262144***

1. Actor saves the script file

**Alternate Workflow:** None

**Post – Condition:** Ansible script is ready to configure New Node

## Use Case 13: Create Ansible Script – To deploy Onboarding

**Actor:** DevOps Member

**Description:** Actor creates Ansible script with the instructions to deploy onboarding on Master and Node Droplet

**Pre – Condition:** Actor can use any software or notepad to write the script

**Normal Workflow:**

1. Actor writes the script to deploy onboarding. Steps involved to deploy onboarding

***---***

***- hosts: master***

***become\_user: centos***

***vars:***

***backend\_uri: https://onboarding-backend.cccsuk.co.uk***

***frontend\_uri: onboarding-frontend.cccsuk.co.uk***

***keycloak\_uri: https://onboarding-keycloak.cccsuk.co.uk***

***billing\_uri: https://onboarding-billing.cccsuk.co.uk***

***jenkinsuri: https://jenkins.cccsuk.co.uk***

***stripepublish\_key: pk\_test\_jvWyZ1jtyG4XVvrAFMUXC4Gj***

***stripesecret\_key: sk\_test\_9JaS2xziotPnAM8SDymuwXyN***

***stripetaxpercent: 20***

***Clientname: CC\_app***

***tasks:***

***- name: Copy the deployment file***

***copy:***

***src: keycloak***

***dest: /opt/onboarding/***

***force: yes***

***- name: Copy the deployment file***

***copy:***

***src: onboarding***

***dest: /opt/onboarding/***

***force: yes***

***- name: Creating a Application for {{ Clientname }}***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create namespace {{ Clientname }}"***

***- name: Deploying Nexus Docker secrets***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create secret docker-registry dockersecrete --docker-server=165.22.121.164:8086 --docker-username=CC\_DevOps --docker-password=3RiNO#A\_ --namespace={{ Clientname }}"***

***- name: Deploying keycloak db Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/onboarding/keycloak/keycloakdb/ --namespace={{ Clientname }}"***

***- name: Setting up DB***

***command: "sleep 5" # wait 10 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying keycloak Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/onboarding/keycloak/ --namespace={{ Clientname }}"***

***- name: Deploying onboarding backend db Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/onboarding/onboarding/onboarding\_db/ --namespace={{ Clientname }}"***

***- name: Wait for 10 sec***

***command: "sleep 10" # wait 10 sec bcs the db needs to be up before connecting to backend***

***- name: creating onboarding frontend configmap***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create configmap frontend-config --from-literal REACT\_APP\_BACKEND\_URI={{ backend\_uri }}/ --from-literal REACT\_APP\_BILLING\_BACKEND\_URI={{ billing\_uri }}/ --from-literal REACT\_APP\_STRIPE\_PUBLISH\_KEY={{ stripepublish\_key }} --namespace={{ Clientname }}"***

***- name: creating onboarding backend configmap***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create configmap backendurl-config --from-literal FRONTENDURI=https://{{ frontend\_uri }}/ --from-literal keycloakURI={{ keycloak\_uri }}/auth/ --namespace={{ Clientname }}"***

***- name: creating onboarding billing configmap***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create configmap billingurl-config --from-literal keycloakURI={{ keycloak\_uri }}/auth/ --from-literal STRIPE\_API\_KEY={{ stripesecret\_key }} --from-literal BILLING\_BACKEND\_URI={{ billing\_uri }}/ --from-literal ONBOARD\_BACKEND\_URI={{ backend\_uri }} --from-literal STRIPE\_PUBLISH\_KEY={{ stripepublish\_key }} --from-literal STRIPE\_TAX\_PERCENT={{ stripetaxpercent }} --from-literal JENKINSURI={{ jenkinsuri }}/ --namespace={{ Clientname }}"***

***- name: Deploying onboarding Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/onboarding/onboarding/onboarding\_app/ --namespace={{ Clientname }}"***

***- name: Deploying onboarding Billing DB Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/onboarding/onboarding/onboarding\_billing\_db/ --namespace={{ Clientname }}"***

***- name: Deploying onboarding Billing backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/onboarding/onboarding/onboarding\_billing\_app/ --namespace={{ Clientname }}"***

***- name: Deploying onboarding services Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/onboarding/onboarding/app\_services/ --namespace={{ Clientname }}"***

***- name: Deploying Application stack***

***command: "sleep 6m" # wait 6 mins bcs the db needs to be up before connecting to backend***

***- name: Deploying onboarding Backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment backend --replicas=0 --namespace={{ Clientname }}"***

***- name: connecting to backend Db***

***command: "sleep 20" # wait 20 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying onboarding Backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment backend --replicas=1 --namespace={{ Clientname }}"***

***- name: Wait for 50 sec***

***command: "sleep 50" # wait 50 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying onboarding Backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment backend --replicas=0 --namespace={{ Clientname }}"***

***- name: Connecting the Application stack***

***command: "sleep 20" # wait 20 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying onboarding Backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment backend --replicas=1 --namespace={{ Clientname }}"***

***- name: connecting to backend Db***

***command: "sleep 20" # wait 20 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying Billing Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment billing --replicas=0 --namespace={{ Clientname }}"***

***- name: connecting to backend Db***

***command: "sleep 20" # wait 20 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying Billing Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment billing --replicas=1 --namespace={{ Clientname }}"***

***- name: Deploying Billing Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment billing --replicas=0 --namespace={{ Clientname }}"***

***- name: connecting to backend Db***

***command: "sleep 20" # wait 20 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying Billing Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment billing --replicas=1 --namespace={{ Clientname }}"***

1. Actor saves the script file

**Alternate Workflow:** None

**Post – Condition:** Ansible script is ready to deploy onboarding on Master and Node Droplet

## Use Case 14: Create Ansible Script – To deploy Platform

**Actor:** DevOps Member

**Description:** Actor creates Ansible script with the instructions to deploy platform on Master and Node Droplet

**Pre – Condition:** Actor can use any software or notepad to write the script

**Normal Workflow:**

1. Actor writes the script to deploy platform. Steps involved to deploy platform

***---***

***- hosts: master***

***become\_user: centos***

***vars:***

***backend\_uri: https://platform-backend.cccsuk.co.uk***

***frontend\_uri: platform-frontend.cccsuk.co.uk***

***keycloak\_uri: https://platform-backend.cccsuk.co.uk***

***elasticfrontenduri: https://elastic-backend.cccsuk.co.uk***

***elasticbackenduri: https://elastic-frontend.cccsuk.co.uk***

***supporturi: https://supporturi.cccseu.com***

***imageuri: https://image.cccseu.com***

***miniouri: https://minio.cccseu.com***

***billinguri: https://platform-demo-billing.cccseu.com***

***stripepublish\_key: pk\_test\_jvWyZ1jtyG4XVvrAFMUXC4Gj***

***stripetaxpercent: 20***

***Clientname: CC\_App***

***tasks:***

***- name: Copy the deployment file***

***copy:***

***src: keycloak***

***dest: /opt/platform/***

***force: yes***

***- name: Copy the deployment file***

***copy:***

***src: platform***

***dest: /opt/platform/***

***force: yes***

***- name: Creating a Application for {{ Clientname }}***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create namespace {{ Clientname }}"***

***- name: Deploying Nexus Docker secrets***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create secret docker-registry dockersecrete --docker-server=165.22.121.164:8086 --docker-username=CC\_DevOps --docker-password=3RiNO#A\_ --namespace={{ Clientname }}"***

***- name: Deploying keycloak db Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/keycloak/keycloakdb/ --namespace={{ Clientname }}"***

***- name: Setting up DB***

***command: "sleep 20" # wait 20 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying keycloak Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/keycloak/ --namespace={{ Clientname }}"***

***- name: Deploying Platform backend db Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/platform/platform\_db/ --namespace={{ Clientname }}"***

***- name: Wait for 50 sec***

***command: "sleep 50" # wait 10 sec bcs the db needs to be up before connecting to backend***

***- name: creating elasticsearch backend configmap***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create configmap elasticsearchbackendurl-config --from-literal ELASTICURI={{ elasticfrontenduri }}/ --namespace={{ Clientname }}"***

***- name: Deploying Platform Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/platform/platform\_app/ --namespace={{ Clientname }}"***

***- name: deploying Elasticsearch App***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/platform/platform\_elasticsearch/ --namespace={{ Clientname }}"***

***- name: deploying support App***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/platform/platform\_support\_app/ --namespace={{ Clientname }}"***

***- name: deploying support mongo db App***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/platform/platform\_support\_db/ --namespace={{ Clientname }}"***

***- name: deploying Imageserver App***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/platform/platform\_imageserver/ --namespace={{ Clientname }}"***

***- name: creating frontend configmap***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create configmap frontend-config --from-literal REACT\_APP\_BACKEND\_URI={{ backend\_uri }}/ --from-literal REACT\_APP\_IMAGE\_BACKEND\_URI={{ imageuri }}/ --from-literal REACT\_APP\_BILLING\_BACKEND\_URI={{ billinguri }} --from-literal REACT\_APP\_STRIPE\_PUBLISH\_KEY={{ stripepublish\_key }} --from-literal STRIPE\_TAX\_PERCENT={{ stripetaxpercent }} --namespace={{ Clientname }}"***

***- name: creating platform backend configmap***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create configmap backendurl-config --from-literal FRONTENDURI=https://{{ frontend\_uri }}/ --from-literal keycloakURI={{ keycloak\_uri }}/auth/ --from-literal ELASTIC\_BACKEND\_URI={{ elasticbackenduri }}/ --from-literal SUPPORT\_API\_URI={{ supporturi }}/api/v1/ --from-literal SUPPORT\_URI={{ supporturi }}/ --namespace={{ Clientname }}"***

***- name: creating imageurl configmap***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf create configmap imageurl-config --from-literal MINIO\_URI={{ miniouri }}/ --namespace={{ Clientname }}"***

***- name: Deploying minio Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/platform/platform\_minio/ --namespace={{ Clientname }}"***

***- name: Deploying service application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f /opt/platform/platform/app\_services/ --namespace={{ Clientname }}"***

***- name: Deploying Application stack***

***command: "sleep 8m" # wait 10 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying Platform Backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment backend --replicas=0 --namespace={{ Clientname }}"***

***- name: connecting to backend Db***

***command: "sleep 20" # wait 10 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying Platform Backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment backend --replicas=1 --namespace={{ Clientname }}"***

***- name: Wait for 50 sec***

***command: "sleep 50" # wait 10 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying Platform Backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment backend --replicas=0 --namespace={{ Clientname }}"***

***- name: Connecting the Application stack***

***command: "sleep 20" # wait 10 sec bcs the db needs to be up before connecting to backend***

***- name: Deploying Platform Backend Application***

***command: "kubectl --kubeconfig /etc/kubernetes/admin.conf scale deployment backend --replicas=1 --namespace={{ Clientname }}"***

1. Actor saves the script file

**Alternate Workflow:** None

**Post – Condition:** Ansible script is ready to deploy platform on Master and Node Droplet

## Use Case 15: Create a Repository in GitHub

**Actor:** DevOps Member

**Description:** Actor creates a repository in GitHub to save the Ansible Script to install Jenkins and Nexus.

**Pre – Condition:** Actor needs to have a GitHub account to Login.

**Normal Workflow:**

1. Actor logins to GitHub using the credentials
2. Actor clicks “+” sign available at the top right near the profile
3. Actor selects “New Repository” from the dropdown
4. Actor redirects to the Repository creation page
5. Actor fills the name for the Repository under “Repository Name”
6. Actor fills the description, if needed
7. Actor clicks “Create Repository”

**Alternate Workflow:**

1. If the Actor provides the wrong credentials
   1. Actor receives an error response
   2. Actor will not be able to login to their account
2. If the repository name field is empty
   1. “Create Repository” button will not be enabled

**Post – Condition:** Actor can view the created repository under their account

## Use Case 16: Push Ansible Scripts to the Repository

**Actor:** DevOps Member

**Description:** Actor pushes the created Ansible scripts inside the GitHub repository.

**Pre – Condition:** Actor needs to have the Ansible Scripts at their end and also have a repository in GitHub

**Normal Workflow:**

1. Actor navigates to the main page of the repository
2. Actor clicks “Upload files”
3. Actor drag and drop the Shell Script into the repository
4. Actor add a description about the changes under “Commit Changes”, if needed
5. Actor selects “Commit directly to the master branch” or “Create a new branch” to commit
6. Actor clicks “Commit changes”

**Alternate Workflow:** None

**Post – Condition:** Ansible scripts saved into the repository

## Use Case 17: Install Ansible – Job Droplet

**Actor:** DevOps Member

**Description:** Actor installs Ansible inside the Droplet to run the Ansible scripts created for the installation of Jenkins and Nexus.

**Pre – Condition:** Actor connects with the Job server via terminal

**Normal Workflow:**

Actor installs the CentOS 7 EPEL (Extra Packages for Enterprise Linux)

***sudo yum install -y epel-release***

Actor installs Ansible

***sudo yum install -y ansible***

**Alternate Workflow:** None

**Post – Condition:** Ansible installed inside the Job Droplet

## Use Case 18: Install Git Client – Job Droplet

**Actor:** DevOps Member

**Description:** Actor installs Git Client inside the Droplet to run the Ansible scripts created for the installation of Jenkins and Nexus.

**Pre – Condition:** Actor connects with the Job server via terminal

**Normal Workflow:**

Actor installs Git Client

***sudo yum install -y git***

Actor checks the version of the Git Client installed

***git --version***

**Alternate Workflow:** None

**Post – Condition:** Git Client gets installed inside the Job Droplet

## Use Case 19: Run Jenkin Installation Ansible Script – Job Droplet

**Actor:** DevOps Member

**Description:** Once Actor installs Ansible, they will get the access to run the Ansible scripts by cloning the GitHub repository. Run the Jenkins Installation Ansible Script

**Pre – Condition:** Actor connects with the Job server via terminal

**Normal Workflow:**

1. Actor clones the GitHub repository inside the Droplet which contains Ansible script

***git clone*** [***https://github.com/ComplianceCompedium/CCS-Onboarding-DevOps.git***](https://github.com/ComplianceCompedium/CCS-Onboarding-DevOps.git)

1. Actor gets into the folder which contains the Ansible Script

***cd CCS-Onboarding-DevOps***

1. Actor changes the branch to Development from Master

***git checkout development***

1. Actor gets in the Centos folder

***cd Centos/Ansible\_jenkins***

1. Actor runs the Ansible Script of Jenkins

***ansible-playbook jenkins.yaml***

**Alternate Workflow:** None

**Post – Condition:** Jenkins Installation completed successfully inside the Job Droplet

## Ue Case 20: Create User in Jenkins

**Actor:** DevOps Member

**Description:** Actor needs to create user in Jenkins to write the Jenkins job

**Pre – Condition:** Jenkins needs to be installed in the Job Droplet.

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor view the details to Unlock Jenkins
3. Actor opens the terminal
4. Actor logins into the Job Droplet server

***ssh root@<Job Droplet IP>***

1. Actor copies the path given in the Unlock Jenkins
2. Actor gets the Administrator password by pasting the copied path along with the command

***cat /var/jenkins\_home/secrets/initialAdminPassword***

1. Actor receives the password in the terminal
2. Actor copies the password
3. Actor goes to the browser and pastes the copied password under “Administrator Password”
4. Actor clicks “Continue”
5. Actor clicks “Install Suggested Plugins”
6. Actor redirects to the “Create First Admin User” Screen
7. Actor fills the username
8. Actor fills the password *Avoid symbols in the password*
9. Actor fills the confirm password *Avoid symbols in the password*
10. Actor fills the full name
11. Actor fills the email address
12. Actor clicks “Save & Finish”
13. Actor views the Jenkins URL
14. Actor clicks “Save & Finish”
15. Actor views the Getting started section
16. Actor clicks “Start Using Jenkins

**Alternate Workflow:**

1. If the copy is not properly done
   1. Actor will receive an error message

If the username field is empty

Actor receives an error message

If the password field is empty

Actor receives an error message

If the confirm password field is empty

Actor receives an error message

1. If the confirm password not matches with the password

Actor receives an error message

1. If the full name field is empty
   1. Actor receives an error message
2. If the email address field is empty
   1. Actor receives an error message
3. If the Email address is not valid
   1. Actor receives an error message

**Post – Condition:** Jenkins User gets created

## Use Case 21: Create Credentials in Jenkins for GitHub

**Actor:** DevOps Member

**Description:** Actor needs to create credentials in Jenkins for GitHub before writing the Jenkins job

**Pre – Condition:** Jenkins needs to be installed in the Job Droplet.

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the user credentials and logs in
3. Actor clicks “Credentials” in the side menu
4. Actor clicks “Jenkins” under “Stores scoped to Jenkins”
5. Actor clicks “Global Credentials” under “System”
6. Actor clicks “Adding some credentials” under “Global Credentials”
7. Actor fills the username of GitHub
8. Actor fills the password of GitHub
9. Actor fills the ID as per their expectations
10. Actor clicks “Ok”

**Alternate Workflow:** None

**Post – Condition:** Credentials created successfully

## Use Case 22: Create Jenkins Job – To run Onboarding Front End Script

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Onboarding Front End

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Onboarding Front End URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Onboarding Front End Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Onboarding Front End

## Use Case 23: Create Jenkins Job – To run Onboarding Back End Script

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Onboarding Back End

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Onboarding Back End URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Onboarding Back End Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Onboarding Back End

## Use Case 24: Create Jenkins Job – To run Onboarding Key Cloak

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Onboarding Key Cloak

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Onboarding Key Cloak URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Onboarding Key Cloak Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Onboarding Key Cloak

## Use Case 25: Create Jenkins Job – To run Onboarding Billing

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Onboarding Billing

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Onboarding Billing URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Onboarding Billing Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Onboarding Billing

## Use Case 26: Create Jenkins Job – To run Platform Front End

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Platform Front End

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Platform Front End URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Platform Front End Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Platform Front End

## Use Case 27: Create Jenkins Job – To run Platform Back End

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Platform Back End

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Platform Back End URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Platform Back End Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Platform Back End

## Use Case 28: Create Jenkins Job – To run Platform Key Cloak

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Platform Key Cloak

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Platform Key Cloak URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Platform Keycloak Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Platform Key Cloak

## Use Case 29: Create Jenkins Job – To run Platform Elastic Search

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Platform Elastic Search

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Platform Elastic Search URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Platform Elastic Search Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Platform Elastic Search

## Use Case 30: Create Jenkins Job – To run Platform Image Service

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the script of Platform Image Service

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as per their expectation
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
8. Actor selects “Git” from SCM dropdown
9. Actor pastes the URL of the Platform Image Service URL of the GitHub Repository
10. Actor selects the credentials from the dropdown which we created
11. Actor adds the branch specifier as “\*/development” under “Branches to build”
12. Actor fills the script path of the Platform Image Service Jenkins file which is available in GitHub repository
13. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the Platform Image Service

## Use Case 31: Create Jenkins Job – To run Nexus Installation Ansible Script

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the ansible script of Nexus Installation

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “This Project is parameterized” under General
8. Actor selects “String Parameter” under “Add parameter” dropdown
9. Actor fills the name for the String Parameter as *Master\_ip*
10. Actor fills the description for the String Parameter as *Master Ip address*
11. Actor selects “Password Parameter” under “Add parameter” dropdown
12. Actor fills the name for the Password Parameter as *Master\_pass*
13. Actor fills the description for the Password Parameter as *Master passowrd*
14. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
15. Actor selects “Git” from SCM dropdown
16. Actor pastes the URL of the DevOps URL of the GitHub Repository
17. Actor adds the credentials of the GitHub repository
18. Actor adds the branch specifier as “\*/development” under “Branches to build”
19. Actor fills the script path of the Nexus Jenkins file which is available in GitHub repository

***Centos/Ansible\_nexus/Jenkinsfile***

1. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the ansible script of Nexus Installation

## Use Case 32: Create Jenkins Job – To run SSL certification copy Ansible script

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the ansible script of copying the SSL certification

**Pre– Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “This Project is parameterized” under General
8. Actor selects “String Parameter” under “Add parameter” dropdown
9. Actor fills the name for the String Parameter as *Worker\_ip*
10. Actor fills the description for the String Parameter as *Enter the worker IP address*
11. Actor selects “Password Parameter” under “Add parameter” dropdown
12. Actor fills the name for the Password Parameter as *Worker\_pass*
13. Actor fills the description for the Password Parameter as *Enter worker password*
14. Actor selects “Choice Parameter” under “Add parameter” dropdown
15. Actor fills the name for the Choice Parameter as *folder\_name*
16. Actor fills the Choice for the Choice Parameter as *cccseu.com*
17. Actor fills the description for the Choice Parameter as *Choose the folder name*
18. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
19. Actor selects “Git” from SCM dropdown
20. Actor pastes the URL of the DevOps URL of the GitHub Repository
21. Actor adds the credentials of the GitHub repository
22. Actor adds the branch specifier as “\*/development” under “Branches to build”
23. Actor fills the script path of the SSL Jenkins file which is available in GitHub repository

***Centos/wildcard\_ssl/Jenkinsfile***

1. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the ansible script of SSL Certification Copy

## Use Case 33: Create Jenkins Job – To run Ansible scripts of Kubernetes and Docker Installation with Configuration & Nexus Configuration

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the ansible script of Kubernetes and Docker Installation with Kubernetes, Docker and Nexus Configuration.

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “This Project is parameterized” under General
8. Actor selects “String Parameter” under “Add parameter” dropdown
9. Actor fills the name for the String Parameter as *Master\_ip*
10. Actor fills the description for the String Parameter as *Master Ip address*
11. Actor selects “String Parameter” under “Add parameter” dropdown
12. Actor fills the name for the String Parameter as *Worker\_ip*
13. Actor fills the description for the String Parameter as *Worker Ip address*
14. Actor selects “Password Parameter” under “Add parameter” dropdown
15. Actor fills the name for the Password Parameter as *Master\_pass*
16. Actor fills the description for the Password Parameter as *Master passowrd*
17. Actor selects “Password Parameter” under “Add parameter” dropdown
18. Actor fills the name for the Password Parameter as *Worker\_pass*
19. Actor fills the description for the Password Parameter as *worker Passowrd*
20. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
21. Actor selects “Git” from SCM dropdown
22. Actor pastes the URL of the DevOps URL of the GitHub Repository
23. Actor adds the credentials of the GitHub repository
24. Actor adds the branch specifier as “\*/development” under “Branches to build”
25. Actor fills the script path of the Kubernetes Jenkins file which is available in GitHub repository

***Centos/Ansible\_Kubernetes /Jenkinsfile***

1. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the ansible script of Kubernetes and Docker Installation with Kubernetes, Docker and Nexus Configuration.

## Use Case 34: Create Jenkins Job – To run Ansible scripts of New Node Configuration

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the ansible script of New Node Configuration.

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “This Project is parameterized” under General
8. Actor selects “String Parameter” under “Add parameter” dropdown
9. Actor fills the name for the String Parameter as *Master\_ip*
10. Actor fills the description for the String Parameter as *Master Ip address*
11. Actor selects “String Parameter” under “Add parameter” dropdown
12. Actor fills the name for the String Parameter as *Worker\_ip*
13. Actor fills the description for the String Parameter as *Worker Ip address*
14. Actor selects “ Password Parameter” under “Add parameter” dropdown
15. Actor fills the name for the String Parameter as *Master\_pass*
16. Actor fills the description for the String Parameter as *Master passowrd*
17. Actor selects “ Password Parameter” under “Add parameter” dropdown
18. Actor fills the name for the String Parameter as *Worker\_pass*
19. Actor fills the description for the String Parameter as *worker Passowrd*
20. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
21. Actor selects “Git” from SCM dropdown
22. Actor pastes the URL of the DevOps URL of the GitHub Repository
23. Actor adds the credentials of the GitHub repository
24. Actor adds the branch specifier as “\*/development” under “Branches to build”
25. Actor fills the script path of the new node Jenkins file which is available in GitHub repository

***Centos/Adding\_new\_node/Jenkinsfile***

1. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the ansible script of New Node Configuration.

## Use Case 35: Create Jenkins Job – To Mount DNS to IP

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the GoDaddy API which mounts the DNS to IP

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name
5. Actor selects “Freestyle Project”
6. Actor clicks “Ok”
7. Actor selects “This Project is parameterized” under General
8. Actor selects “String Parameter” under “Add parameter” dropdown
9. Actor fills the name for the String Parameter as *domain\_name*
10. Actor fills the Default Value for the String Parameter as *cccseu.com*
11. Actor fills the description for the String Parameter as *Enter the Domain Name*
12. Actor selects “String Parameter” under “Add parameter” dropdown
13. Actor fills the name for the String Parameter as *subdomain\_name*
14. Actor fills the description for the String Parameter as *Enter the subdomain name*
15. Actor selects “String Parameter” under “Add parameter” dropdown
16. Actor fills the name for the String Parameter as *subdomain\_ip*
17. Actor fills the description for the String Parameter as *Enter the subdomain IP*
18. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
19. Actor selects “Git” from SCM dropdown
20. Actor pastes the URL of the DevOps URL of the GitHub Repository
21. Actor adds the credentials of the GitHub repository
22. Actor adds the branch specifier as “\*/development” under “Branches to build”
23. Actor fills the script path of the new node Jenkins file which is available in GitHub repository

***Centos/wildcard\_subdomain/Jenkinsfile***

1. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the GoDaddy API which mounts the DNS to IP.

## Use Case 36: Create Jenkins Job – To run Ansible script of Onboarding Deployment

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the ansible script of Onboarding Deployment.

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “This Project is parameterized” under General
8. Actor selects “String Parameter” under “Add parameter” dropdown
9. Actor fills the name for the String Parameter as *subdomainName*
10. Actor fills the description for the String Parameter as *Enter the Frontend Subdomian*
11. Actor selects “String Parameter” under “Add parameter” dropdown
12. Actor fills the name for the String Parameter as *onboardingbackenduri*
13. Actor fills the description for the String Parameter as *Enter the Backend URL*
14. Actor selects “String Parameter” under “Add parameter” dropdown
15. Actor fills the name for the String Parameter as *keycloakuri*
16. Actor fills the description for the String Parameter as *Enter the Keycloak URL*
17. Actor selects “String Parameter” under “Add parameter” dropdown
18. Actor fills the name for the String Parameter as *billinguri*
19. Actor fills the description for the String Parameter as *BillingURL*
20. Actor selects “String Parameter” under “Add parameter” dropdown
21. Actor fills the name for the String Parameter as *jenkinsuri*
22. Actor fills the description for the String Parameter as *Enter the Jenkins URL*
23. Actor selects “String Parameter” under “Add parameter” dropdown
24. Actor fills the name for the String Parameter as *masterIP*
25. Actor fills the description for the String Parameter as *Master Ip address*
26. Actor selects “Password Parameter” under “Add parameter” dropdown
27. Actor fills the name for the Password Parameter as *masterPsw*
28. Actor fills the description for the Password Parameter as *Worker Ip address*
29. Actor selects “ String Parameter” under “Add parameter” dropdown
30. Actor fills the name for the String Parameter as *nodeIP*
31. Actor fills the description for the String Parameter as *Master passowrd*
32. Actor selects “ Password Parameter” under “Add parameter” dropdown
33. Actor fills the name for the Password Parameter as *nodePsw*
34. Actor fills the description for the Password Parameter as *worker Passowrd*
35. Actor selects “ Password Parameter” under “Add parameter” dropdown
36. Actor fills the name for the Password Parameter as *stripepublish\_key*
37. Actor fills the description for the Password Parameter as *Enter the stripe\_publish\_key*
38. Actor selects “ Password Parameter” under “Add parameter” dropdown
39. Actor fills the name for the Password Parameter as *stripesecret\_key*
40. Actor fills the description for the Password Parameter as *Enter the stripe\_secret\_key*
41. Actor selects “String Parameter” under “Add parameter” dropdown
42. Actor fills the name for the String Parameter as *stripetaxpercent*
43. Actor fills the description for the String Parameter as *Enter the stripe\_tax\_percent*
44. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
45. Actor selects “Git” from SCM dropdown
46. Actor pastes the URL of the DevOps URL of the GitHub Repository
47. Actor adds the credentials of the GitHub repository
48. Actor adds the branch specifier as “\*/development” under “Branches to build”
49. Actor fills the script path of the Onboarding Jenkins file which is available in GitHub repository

***Onboarding\_Launch/Jenkinsfile***

1. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the ansible script of Onboarding Deployment.

## Use Case 37: Create Jenkins Job – To run Ansible script of Platform Deployment

**Actor:** DevOps Member

**Description:** Actor creates Jenkins job to run the ansible script of Platform Deployment.

**Pre – Condition:** Actor needs to have a user to create the Jenkins Job

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials to login
3. Actor clicks “New Item” in the side menu
4. Actor fills the job name as ***“platformdeploy”***
5. Actor selects “Pipeline”
6. Actor clicks “Ok”
7. Actor selects “This Project is parameterized” under General
8. Actor selects “String Parameter” under “Add parameter” dropdown
9. Actor fills the name for the String Parameter as *subdomainName*
10. Actor fills the description for the String Parameter as *Enter the Frontend Subdomian*
11. Actor selects “String Parameter” under “Add parameter” dropdown
12. Actor fills the name for the String Parameter as *platfrombackenduri*
13. Actor fills the description for the String Parameter as *Enter the Backend URL*
14. Actor selects “String Parameter” under “Add parameter” dropdown
15. Actor fills the name for the String Parameter as *keycloakuri*
16. Actor fills the description for the String Parameter as *Enter the Keycloak URL*
17. Actor selects “String Parameter” under “Add parameter” dropdown
18. Actor fills the name for the String Parameter as *elasticfrontenduri*
19. Actor fills the description for the String Parameter as *Enter the elasticfrontend URL*
20. Actor selects “String Parameter” under “Add parameter” dropdown
21. Actor fills the name for the String Parameter as *elasticbackenduri*
22. Actor fills the description for the String Parameter as *Enter the elasticbackend URL*
23. Actor selects “String Parameter” under “Add parameter” dropdown
24. Actor fills the name for the String Parameter as *supporturi*
25. Actor fills the description for the String Parameter as *Enter the support URL*
26. Actor selects “String Parameter” under “Add parameter” dropdown
27. Actor fills the name for the String Parameter as *imageuri*
28. Actor fills the description for the String Parameter as *Enter the image URL*
29. Actor selects “String Parameter” under “Add parameter” dropdown
30. Actor fills the name for the String Parameter as *miniouri*
31. Actor fills the description for the String Parameter as *Enter the minio URL*
32. Actor selects “String Parameter” under “Add parameter” dropdown
33. Actor fills the name for the String Parameter as *billinguri*
34. Actor fills the description for the String Parameter as *Enter the Billing URL*
35. Actor selects “String Parameter” under “Add parameter” dropdown
36. Actor fills the name for the String Parameter as *onboardingbackenduri*
37. Actor fills the description for the String Parameter as *Enter the Onboard Backend URL*
38. Actor selects “String Parameter” under “Add parameter” dropdown
39. Actor fills the name for the String Parameter as *userName*
40. Actor fills the description for the String Parameter as *Enter User name*
41. Actor selects “String Parameter” under “Add parameter” dropdown
42. Actor fills the name for the String Parameter as *email*
43. Actor fills the description for the String Parameter as *Enter user email*
44. Actor selects “String Parameter” under “Add parameter” dropdown
45. Actor fills the name for the String Parameter as *kUserID*
46. Actor fills the description for the String Parameter as *Enter KUID*
47. Actor selects “String Parameter” under “Add parameter” dropdown
48. Actor fills the name for the String Parameter as *stripepublish\_key*
49. Actor fills the description for the String Parameter as *Enter the stripe\_publish\_key*
50. Actor selects “String Parameter” under “Add parameter” dropdown
51. Actor fills the name for the String Parameter as *stripetaxpercent*
52. Actor fills the description for the String Parameter as *Enter the stripe\_tax\_percent*
53. Actor selects “String Parameter” under “Add parameter” dropdown
54. Actor fills the name for the String Parameter as *adminemail*
55. Actor fills the Default Value for the String Parameter as *vignesh.k@akonisoftwareservices.com*
56. Actor fills the description for the String Parameter as *Admin url for ssl creation*
57. Actor selects “String Parameter” under “Add parameter” dropdown
58. Actor fills the name for the String Parameter as *masterIP*
59. Actor fills the Default Value for the String Parameter ***“CURRENT IP OF THE KUBERNETES MASTER”***
60. Actor fills the description for the String Parameter as *Master Ip address*
61. Actor selects “Password Parameter” under “Add parameter” dropdown
62. Actor fills the name for the Password Parameter as *masterPsw*
63. Actor fills the Default Value for the Password Parameter ***“CURRENT PASSWORD OF THE KUBERNETES MASTER”***
64. Actor fills the description for the Password Parameter as *Master password*
65. Actor selects “String Parameter” under “Add parameter” dropdown
66. Actor fills the name for the String Parameter as *nodeIP*
67. Actor fills the Default Value for the String Parameter ***“CURRENT IP OF THE PLATFORM NODE”***
68. Actor fills the description for the String Parameter as *Worker Ip address*
69. Actor selects “Password Parameter” under “Add parameter” dropdown
70. Actor fills the name for the Password Parameter as *nodePsw*
71. Actor fills the Default Value for the Password Parameter ***“CURRENT PASSWORD OF THE PLATFORM NODE”***
72. Actor fills the description for the Password Parameter as *worker Password*
73. Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
74. Actor selects “Git” from SCM dropdown
75. Actor pastes the URL of the DevOps URL of the GitHub Repository
76. Actor adds the credentials of the GitHub repository
77. Actor adds the branch specifier as “\*/development” under “Branches to build”
78. Actor fills the script path of the platform Jenkins file which is available in GitHub repository

***Platform\_Launch/Jenkinsfile***

1. Actor clicks “Apply” & “Save”

**Alternate Workflow:** None

**Post – Condition:** Jenkins Job is ready to run the ansible script of Platform Deployment.

## Use Case 38: Run Nexus Installation Jenkins Job

**Actor:** DevOps Member

**Description:** After the process of Jenkins completed, Actor installs Nexus inside the Nexus Droplet by running the Jenkins Job

**Pre – Condition:** Actor connects with the Job server via terminal

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Nexus Installation Jenkins Job
4. Actor clicks “Build with Parameters” in the side menu
5. Actor fills the Ip and password of the Nexus Droplet
6. DevOps Member clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Nexus gets installed successfully inside Nexus Droplet

## Use Case 39: Configure Nexus – Nexus Droplet

**Actor:** DevOps Member

**Description:** Actor configure Nexus inside the Nexus Droplet.

**Pre – Condition:** Actor connects with the Nexus server via terminal

**Normal Workflow:**

1. Actor enters <Nexus Droplet IP>:8081 in browser
2. Actor sets the new password for the account
3. Actor clicks “Create Repository”
4. Actor fills the Name of the Repository
5. Actor checks and fills the HTTP port number as 8086
6. Actor checks the “Force Basic Authentication”
7. Actor checks the “Enable Docker V1 API” under “Docker Registry API Support”
8. Actor clicks “Save”
9. Actor clicks “Roles” under “Security” in the side menu
10. Actor clicks “Create Role”
11. Actor fills the Role ID
12. Actor fills the Role Description
13. Actor sets the privileges

***nx-repository-admin-docker-CCS-edit***

***nx-repository-admin-docker-CCS-read***

1. Actor clicks “Save”
2. Actor clicks “Users” under “Security” in the side menu
3. Actor clicks “Create local user”
4. Actor fills the ID
5. Actor fills the first name
6. Actor fills the last name
7. Actor fills the email address
8. Actor sets the status as “Active”
9. Actor sets the Role as “nx-admin”
10. Actor clicks “Save”
11. Actor clicks “Create local user” to create another user for Jenkins
12. Actor fills the ID as “Jenkins”
13. Actor fills the first name
14. Actor fills the last name
15. Actor fills the email address
16. Actor sets the status as “Disabled”
17. Actor sets the Role as “Jenkins”
18. Actor clicks “Save”
19. Actor clicks “Anonymous” under “Security” in the side menu
20. Actor unchecks “Allow Anonymous users to access the server”
21. Actor clicks “Save”

**Alternate Workflow:** None

**Post – Condition:** Nexus configured successfully inside the Nexus Droplet

## Use Case 40: Create Credentials in Jenkins for Nexus

**Actor:** DevOps Member

**Description:** Actor needs to create credentials in Jenkins for GitHub before writing the Jenkins job

**Pre – Condition:** Jenkins needs to be installed in the Job Droplet.

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the user credentials and logs in
3. Actor clicks “Credentials” in the side menu
4. Actor clicks “Jenkins” under “Stores scoped to Jenkins”
5. Actor clicks “Global Credentials” under “System”
6. Actor clicks “Adding some credentials” under “Global Credentials”
7. Actor clicks “Add Credentials” in the side menu
8. Actor fills the username of Nexus
9. Actor fills the password of Nexus
10. Actor fills the ID as “dockerhub”
11. Actor clicks “Ok”

**Alternate Workflow:** None

**Post – Condition:** Credentials created successfully

## Use Case 41: Configure Docker – Job Droplet

**Actor:** DevOps Member

**Description:** Actor configures Docker in the Job Droplet

**Pre – Condition:** Actor connects with the Job server via terminal

**Normal Workflow:**

1. Actor gets into the path where the Ansible Jenkins is situated

***cd CCS-Onboarding-DevOps/Centos/Ansible\_jenkins***

1. Actor opens “daemon.json” file to edit

***vi daemon.json***

1. Actor updates the Nexus IP
2. Actor saves the file and exit

***Press “Esc” button followed by “:wq!” And click “Enter”***

1. Actor opens “inventory” file to edit

***vi inventory***

1. Actor includes the following command in the file

***[localhost]***

***localhost ansible\_host=<Job Droplet IP> ansible\_ssh\_user=root ansible\_ssh\_pass=<Job Droplet Password>***

***[all:vars]***

***ansible\_python\_interpreter=/usr/bin/python***

1. Actor saves the file and exit

***Press “Esc” button followed by “:wq!” And click “Enter”***

1. Actor configures Docker in the Job Droplet using the following command

***ansible-playbook -i inventory docker.yml --extra-vars "nexus\_user=<Nexus Username> nexus\_pwd=<Nexus Password> nexus\_ip=<Nexus Droplet IP> nexus\_docker\_port=8086"***

**Alternate Workflow:** None

**Post – Condition:** Docker configured in the Job Droplet Successfully

## Use Case 42: Update Nexus Credentials – Onboarding Frontend

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Onboarding-FrontEnd”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Onboarding frontend script

## Use Case 43: Update Nexus Credentials – Onboarding Backend

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Onboarding-BackEnd”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Onboarding Backend script

## Use Case 44: Update Nexus Credentials – Onboarding Keycloak

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Onboarding-Keycloak”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Onboarding Keycloak script

## Use Case 45: Update Nexus Credentials – Onboarding Billing

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Billing-BackEnd”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Onboarding Billing script

## Use Case 46: Update Nexus Credentials – Platform Frontend

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Platform-Frontend”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Platform Frontend script

## Use Case 47: Update Nexus Credentials – Platform Backend

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Platform-Backend”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Platform Backend script

## Use Case 48: Update Nexus Credentials – Platform Keycloak

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Platform-Keycloak”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Platform Keycloak script

## Use Case 49: Update Nexus Credentials – Platform Elasticsearch Backend

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-ElasticSearch-Backend”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Platform Elasticsearch Backend script

## Use Case 50: Update Nexus Credentials – Platform Image Service

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in all the developers source.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-ImageService-Backend”
2. Actor updates the branch from Master to Development
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Platform Image Backend script

## Use Case 51: Run Onboarding Front End Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Onboarding Front End Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Onboarding Front End Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Onboarding Front End Jenkins Job completed successfully

## Use Case 52: Run Onboarding Back End Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Onboarding Back End Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Onboarding Back End Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Onboarding Back End Jenkins Job completed successfully

## Use Case 53: Run Onboarding Key Cloak Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Onboarding Key Cloak Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Onboarding Key Cloak Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Onboarding Key Cloak Jenkins Job completed successfully

## Use Case 54: Run Onboarding Billing Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Onboarding Billing Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Onboarding Billing Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Onboarding Billing Jenkins Job completed successfully

## Use Case 55: Run Platform Front End Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Platform Front End Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Platform Front End Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Platform Front End Jenkins Job completed successfully

## Use Case 56: Run Platform Back End Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Platform Back End Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Platform Back End Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Platform Back End Jenkins Job completed successfully

## Use Case 57: Run Platform Key Cloak Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Platform Key Cloak Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Platform Key Cloak Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Platform Key Cloak Jenkins Job completed successfully

## Use Case 58: Run Platform Elastic Search Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Platform Elastic Search Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Platform Elastic Search Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Platform Elastic Search Jenkins Job completed successfully

## Use Case 59: Run Platform Image Service Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Platform Image Service Jenkins Job to build the source as an Image and push it over to Nexus Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Platform Image Service Jenkins Job
4. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Platform Image Service Jenkins Job completed successfully

## Use Case 60: Update Nexus Credentials - Kubernetes

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in the Ansible script of Kubernetes

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Onboarding-DevOps”
2. Actor updates the branch from Master to Development
3. Actor clicks “Centos”
4. Actor clicks “Ansible\_Kubernetes”
5. Actor clicks “Jenkinsfile”
6. Actor clicks the Edit icon to edit it
7. Actor updates the Nexus credentials under “Docker Installation” Stage

***ansible-playbook -i inventory docker.yml --extra-vars "nexus\_user=<Nexus Username> nexus\_pwd=<Nexus Password> nexus\_ip=<Nexus Droplet IP> nexus\_docker\_port=8086"***

1. Actor enters the command and clicks ”Commit”
2. Actor clicks “Ansible\_Kubernetes”
3. Actor clicks “daemon.json”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Kubernetes Ansible script

## Use Case 61: Run Kubernetes and Docker Installation with Configuration & Nexus Configuration Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Kubernetes and Docker Jenkins Job to install Kubernetes and Docker. Also, it configures Kubernetes, Docker and Nexus inside the Master and Onboarding Node Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Kubernetes Jenkins Job
4. Actor clicks “Build with Parameters” in the side menu
5. Actor fills the IP and password of the Master & Onboarding Node Droplet
6. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Kubernetes and Docker Installation with Kubernetes, Docker and Nexus Configuration completed successfully inside master and onboarding node Droplet

## Use Case 62: Update Nexus Credentials – New Node Kubernetes

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials in the Ansible script of Kubernetes

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Onboarding-DevOps”
2. Actor updates the branch from Master to Development
3. Actor clicks “Centos”
4. Actor clicks “Adding\_new\_node”
5. Actor clicks “Jenkinsfile”
6. Actor clicks the Edit icon to edit it
7. Actor updates the Nexus credentials under “Docker Installation” Stage

***ansible-playbook -i inventory docker.yml --extra-vars "nexus\_user=<Nexus Username> nexus\_pwd=<Nexus Password> nexus\_ip=<Nexus Droplet IP> nexus\_docker\_port=8086"***

1. Actor enters the command and clicks ”Commit”
2. Actor clicks “Adding\_new\_node”
3. Actor clicks “daemon.json”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus IP
6. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the Kubernetes Ansible script

## Use Case 63: Run Kubernetes and Docker Installation with Configuration & Nexus Configuration Jenkins Job for New Node.

**Actor:** DevOps Member

**Description:** Actor runs the Kubernetes and Docker Jenkins Job to install Kubernetes and Docker. Also, it configures Kubernetes, Docker and Nexus inside the Master and Onboarding Node Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the New\_Node\_Kubernetes Jenkins Job
4. Actor clicks “Build with Parameters” in the side menu
5. Actor fills the IP and password of the Master & Platform Node Droplet
6. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Kubernetes and Docker Installation with Kubernetes, Docker and Nexus Configuration completed successfully inside master and Platform node Droplet

## Use Case 64: Run SSL certification copy Jenkins Job - Onboarding

**Actor:** DevOps Member

**Description:** Actor runs the SSL Certification copy Jenkins Job to copy the SSL certification from GitHub to the Onboarding Node Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the SSL Jenkins Job
4. Actor clicks “Build with Parameters” in the side menu
5. Actor fills the IP and password of the Onboarding Node Droplet
6. Actor choose the certificate name under “folder\_name” dropdown as per their expectations
7. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** SSL certification successfully copied from GitHub to onboarding node Droplet

## Use Case 65: Run SSL certification copy Jenkins Job - Platform

**Actor:** DevOps Member

**Description:** Actor runs the SSL Certification copy Jenkins Job to copy the SSL certification from GitHub to the Platform Node Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the SSL Jenkins Job
4. Actor clicks “Build with Parameters” in the side menu
5. Actor fills the IP and password of the Platform Node Droplet
6. Actor choose the certificate name under “folder\_name” dropdown as per their expectations
7. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** SSL certification successfully copied from GitHub to platform node Droplet

## Use Case 66: Run DNS mount Jenkins Job - Onboarding

**Actor:** DevOps Member

**Description:** Actor runs the DNS Mount Jenkins Job to mount the onboarding node droplet IP to DNS.

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the DNS Jenkins Job
4. Actor clicks “Build with Parameters” in the side menu
5. Actor fills the domain name
6. Actor fills the Subdomain name with Onboarding Front End domain
7. Actor fills the IP of the Onboarding Node Droplet
8. Actor clicks “Build”
9. Actor repeats the above process for Onboarding back end, Onboarding Keycloak and Onboarding Billing where the sub domain names will be different for all the process

**Alternate Workflow:** None

**Post – Condition:** Onboarding Node Droplet IP mounted successfully to DNS

## Use Case 67: Run DNS mount Jenkins Job - Platform

**Actor:** DevOps Member

**Description:** Actor runs the DNS Mount Jenkins Job to mount the Platform Node Droplet IP to DNS.

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the DNS Jenkins Job
4. Actor clicks “Build with Parameters” in the side menu
5. Actor fills the domain name
6. Actor fills the Subdomain name with “\*”
7. Actor fills the IP of the Platform Node Droplet
8. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Platform Node Droplet IP mounted successfully to DNS

## Use Case 68: Update Nexus Credentials with version updates – Onboarding Deployment

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials with the version update before deploy the onboarding.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Onboarding-DevOps”
2. Actor updates the branch from Master to Development
3. Actor clicks “Onboarding\_Launch”
4. Actor clicks “onboarding\_keycloak\_fullstack”
5. Actor clicks “App\_stack\_ansible.yaml”
6. Actor clicks the Edit icon to edit it
7. Actor updates the Nexus credentials in the command under “Deploying Nexus Docker secrets” name

***"kubectl --kubeconfig /etc/kubernetes/admin.conf create secret docker-registry dockersecrete --docker-server=<Nexus Droplet IP>:8086 --docker-username=<Nexus Username> --docker-password=<Nexus Password> --namespace={{ Clientname }}"***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “onboarding\_keycloak\_fullstack” by clicking it
3. Actor clicks “Keycloak”
4. Actor clicks “keycloak\_deploy.yaml”
5. Actor clicks the Edit icon to edit it
6. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/onboard\_keycloak:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “onboarding\_keycloak\_fullstack” by clicking it
3. Actor clicks “Onboarding”
4. Actor clicks “onboarding\_app”
5. Actor clicks “backend\_deployment.yaml”
6. Actor clicks the Edit icon to edit it
7. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/onboard\_backend:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “onboarding\_app” by clicking it
3. Actor clicks “frontent\_deploy.yaml”
4. Actor clicks the Edit icon to edit it
5. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/onboard\_frontend:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “Onboarding” by clicking it
3. Actor clicks “onboarding\_billing\_app”
4. Actor clicks “billing\_deploy.yaml”
5. Actor clicks the Edit icon to edit it
6. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/onboard\_billing:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the scripts of onboarding

## Use Case 69: Update Nexus Credentials with version updates – Platform Deployment

**Actor:** DevOps Member

**Description:** Actor updates the nexus credentials with the version update before deploy the platform.

**Pre – Condition:** Actor needs to login to GitHub

**Normal Workflow:**

1. Actor clicks “CCS-Onboarding-DevOps”
2. Actor updates the branch from Master to Development
3. Actor clicks “Platform\_Launch”
4. Actor clicks “Jenkinsfile”
5. Actor clicks the Edit icon to edit it
6. Actor updates the IP of Master and Node Droplet inside the script

***string(name: 'masterIP' , defaultValue: "<Master IP>", description: 'Master Ip address')***

***password(name: 'masterPsw' , defaultValue: "<Master Password>", description: 'Master password')***

***string(name: 'nodeIP' , defaultValue: "<Platform Node IP>", description: 'Worker Ip address')***

***password(name: 'nodePsw' , defaultValue: "<Platform Node Password>", description: "worker Password")***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “Platform\_Launch” by clicking it
3. Actor clicks “platform\_keycloak\_fullstack”
4. Actor clicks “App\_stack\_ansible.yaml”
5. Actor clicks the Edit icon to edit it
6. Actor updates the Nexus credentials in the command under “Deploying Nexus Docker secrets” name

***"kubectl --kubeconfig /etc/kubernetes/admin.conf create secret docker-registry dockersecrete --docker-server=<Nexus Droplet IP>:8086 --docker-username=<Nexus Username> --docker-password=<Nexus Password> --namespace={{ Clientname }}"***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “platform\_keycloak\_fullstack” by clicking it
3. Actor clicks “Keycloak”
4. Actor clicks “keycloak\_deploy.yaml”
5. Actor clicks the Edit icon to edit it
6. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/platform\_keycloak:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “platform\_keycloak\_fullstack” by clicking it
3. Actor clicks “Platform”
4. Actor clicks “platform\_elasticsearch”
5. Actor clicks “elasticbackend\_deployment.yaml”
6. Actor clicks the Edit icon to edit it
7. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/platform\_elasticsearch:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “Platform” by clicking it
3. Actor clicks “platform\_app”
4. Actor clicks “backend\_deployment.yaml”
5. Actor clicks the Edit icon to edit it
6. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/platform\_backend:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “Platform” by clicking it
3. Actor clicks “platform\_app”
4. Actor clicks “frontend\_deploy.yaml”
5. Actor clicks the Edit icon to edit it
6. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/platform\_frontend:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”
2. Actor redirects to “Platform” by clicking it
3. Actor clicks “platform\_imageserver”
4. Actor clicks “imagebackend\_deployment.yaml”
5. Actor clicks the Edit icon to edit it
6. Actor updates the Nexus credentials in the command under “Containers”

***- image: <Nexus Droplet IP>:8086/platform\_image:<Version tag from the Nexus UI>***

1. Actor enters the command and clicks ”Commit”

**Alternate Workflow:** None

**Post – Condition:** Nexus Credentials get updated in the scripts of platform

## Use Case 70: Run Onboarding Deploy Jenkins Job

**Actor:** DevOps Member

**Description:** Actor runs the Onboarding Deploy Jenkins Job to deploy the onboarding script in Master and Onboarding Node Droplet

**Pre – Condition:** Actor needs to log in to Jenkins

**Normal Workflow:**

1. Actor enters <Job Droplet IP>:8080 in browser
2. Actor enters the credentials and logs in
3. Actor clicks the Onboarding Deploy Jenkins Job
4. Actor clicks “Build with Parameters” in the side menu
5. Actor fills the Onboarding frontend url in without “https” and remaining url’s with “https” in parameters fields.
6. Actor clicks “Build”

**Alternate Workflow:** None

**Post – Condition:** Onboarding successfully deployed to Master and Onboarding Node Droplet