Compliance Compendium

USE CASES – DevOps

Compliance Compendium

Table of Contents

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>

   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 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>

   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 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/ComplianceCompendium/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/

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}}

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

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 }}"

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 }}"

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
   Actor clicks “+” sign available at the top right near the profile
   Actor selects “New Repository” from the dropdown
   Actor redirects to the Repository creation page
   Actor fills the name for the Repository under “Repository Name”
   Actor fills the description, if needed
   Actor clicks “Create Repository”

Alternate Workflow:

1a. If the Actor provides the wrong credentials
   Actor receives an error response
   Actor will not be able to login to their account

5a. If the repository name field is empty
   “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
   Actor clicks “Upload files”
   Actor drag and drop the Shell Script into the repository
   Actor add a description about the changes under “Commit Changes”, if needed
   Actor selects “Commit directly to the master branch” or “Create a new branch” to commit
   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

   Actor gets into the folder which contains the Ansible Script

   cd CCS-Onboarding-DevOps

   Actor changes the branch to Development from Master

   git checkout development

   Actor gets in the Centos folder

   cd Centos/Ansible_jenkins

   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

    Actor view the details to Unlock Jenkins

    Actor opens the terminal

    Actor logins into the Job Droplet server

    ssh root@<Job Droplet IP>

    Actor copies the path given in the Unlock Jenkins

    Actor gets the Administrator password by pasting the copied path along with the command

    cat /var/jenkins_home/secrets/initialAdminPassword

    Actor receives the password in the terminal

    Actor copies the password

    Actor goes to the browser and pastes the copied password under “Administrator Password”

    Actor clicks “Continue”

    Actor clicks “Install Suggested Plugins”

    Actor redirects to the “Create First Admin User” Screen

    Actor fills the username

    Actor fills the password *Avoid symbols in the password*

    Actor fills the confirm password *Avoid symbols in the password*

    Actor fills the full name

    Actor fills the email address

    Actor clicks “Save & Finish”

    Actor views the Jenkins URL

    Actor clicks “Save & Finish”

    Actor views the Getting started section

    Actor clicks “Start Using Jenkins

Alternate Workflow:

9a. If the copy is not properly done

    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

18d. If the confirm password not matches with the password

Actor receives an error message

e. If the full name field is empty

Actor receives an error message

f. If the email address field is empty

Actor receives an error message

g. If the Email address is not valid

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
   Actor enters the user credentials and logs in
   Actor clicks “Credentials” in the side menu
   Actor clicks “Jenkins” under “Stores scoped to Jenkins”
   Actor clicks “Global Credentials” under “System”
   Actor clicks “Adding some credentials” under “Global Credentials”
   Actor fills the username of GitHub
   Actor fills the password of GitHub
   Actor fills the ID as per their expectations
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Onboarding Front End URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Onboarding Front End Jenkins file which is available in GitHub repository
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Onboarding Back End URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Onboarding Back End Jenkins file which is available in GitHub repository
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Onboarding Key Cloak URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Onboarding Key Cloak Jenkins file which is available in GitHub repository
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Onboarding Billing URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Onboarding Billing Jenkins file which is available in GitHub repository
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Platform Front End URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Platform Front End Jenkins file which is available in GitHub repository
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Platform Back End URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Platform Back End Jenkins file which is available in GitHub repository
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Platform Key Cloak URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Platform Keycloak Jenkins file which is available in GitHub repository
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Platform Elastic Search URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Platform Elastic Search Jenkins file which is available in GitHub repository
   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name as per their expectation
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the Platform Image Service URL of the GitHub Repository
   Actor selects the credentials from the dropdown which we created
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the Platform Image Service Jenkins file which is available in GitHub repository
   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

   Actor enters the credentials to login

   Actor clicks “New Item” in the side menu

   Actor fills the job name

   Actor selects “Pipeline”

   Actor clicks “Ok”

   Actor selects “This Project is parameterized” under General

   Actor selects “String Parameter” under “Add parameter” dropdown

   Actor fills the name for the String Parameter as *Master_ip*

   Actor fills the description for the String Parameter as *Master Ip address*

   Actor selects “Password Parameter” under “Add parameter” dropdown

   Actor fills the name for the Password Parameter as *Master_pass*

   Actor fills the description for the Password Parameter as *Master passowrd*

   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options

   Actor selects “Git” from SCM dropdown

   Actor pastes the URL of the DevOps URL of the GitHub Repository

   Actor adds the credentials of the GitHub repository

   Actor adds the branch specifier as “*/development” under “Branches to build”

   Actor fills the script path of the Nexus Jenkins file which is available in GitHub repository

   Centos/Ansible_nexus/Jenkinsfile

   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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “This Project is parameterized” under General
   Actor selects “String Parameter” under “Add parameter” dropdown
   Actor fills the name for the String Parameter as *Worker_ip*
   Actor fills the description for the String Parameter as *Enter the worker IP address*
   Actor selects “Password Parameter” under “Add parameter” dropdown
   Actor fills the name for the Password Parameter as *Worker_pass*
   Actor fills the description for the Password Parameter as *Enter worker password*
   Actor selects “Choice Parameter” under “Add parameter” dropdown
   Actor fills the name for the Choice Parameter as *folder_name*
   Actor fills the Choice for the Choice Parameter as *cccseu.com*
   Actor fills the description for the Choice Parameter as *Choose the folder name*
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the DevOps URL of the GitHub Repository
   Actor adds the credentials of the GitHub repository
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the SSL Jenkins file which is available in GitHub repository
   Centos/wildcard_ssl/Jenkinsfile

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

Actor enters the credentials to login

Actor clicks “New Item” in the side menu

Actor fills the job name

Actor selects “Pipeline”

Actor clicks “Ok”

Actor selects “This Project is parameterized” under General

Actor selects “String Parameter” under “Add parameter” dropdown

Actor fills the name for the String Parameter as *Master_ip*

Actor fills the description for the String Parameter as *Master Ip address*

Actor selects “String Parameter” under “Add parameter” dropdown

Actor fills the name for the String Parameter as *Worker_ip*

Actor fills the description for the String Parameter as *Worker Ip address*

Actor selects “Password Parameter” under “Add parameter” dropdown

Actor fills the name for the Password Parameter as *Master_pass*

Actor fills the description for the Password Parameter as *Master passowrd*

Actor selects “Password Parameter” under “Add parameter” dropdown

Actor fills the name for the Password Parameter as *Worker_pass*

Actor fills the description for the Password Parameter as *worker Passowrd*

Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options

Actor selects “Git” from SCM dropdown

Actor pastes the URL of the DevOps URL of the GitHub Repository

Actor adds the credentials of the GitHub repository

Actor adds the branch specifier as “*/development” under “Branches to build”

Actor fills the script path of the Kubernetes Jenkins file which is available in GitHub repository

Centos/Ansible_Kubernetes /Jenkinsfile

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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name
   Actor selects “Pipeline”
   Actor clicks “Ok”
   Actor selects “This Project is parameterized” under General
   Actor selects “String Parameter” under “Add parameter” dropdown
   Actor fills the name for the String Parameter as *Master_ip*
   Actor fills the description for the String Parameter as *Master Ip address*
   Actor selects “String Parameter” under “Add parameter” dropdown
   Actor fills the name for the String Parameter as *Worker_ip*
   Actor fills the description for the String Parameter as *Worker Ip address*
   Actor selects “ Password Parameter” under “Add parameter” dropdown
   Actor fills the name for the String Parameter as *Master_pass*
   Actor fills the description for the String Parameter as *Master passowrd*
   Actor selects “ Password Parameter” under “Add parameter” dropdown
   Actor fills the name for the String Parameter as *Worker_pass*
   Actor fills the description for the String Parameter as *worker Passowrd*
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the DevOps URL of the GitHub Repository
   Actor adds the credentials of the GitHub repository
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the new node Jenkins file which is available in GitHub repository

Centos/Adding_new_node/Jenkinsfile

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
   Actor enters the credentials to login
   Actor clicks “New Item” in the side menu
   Actor fills the job name
   Actor selects “Freestyle Project”
   Actor clicks “Ok”
   Actor selects “This Project is parameterized” under General
   Actor selects “String Parameter” under “Add parameter” dropdown
   Actor fills the name for the String Parameter as *domain_name*
   Actor fills the Default Value for the String Parameter as *cccseu.com*
   Actor fills the description for the String Parameter as *Enter the Domain Name*
   Actor selects “String Parameter” under “Add parameter” dropdown
   Actor fills the name for the String Parameter as *subdomain_name*
   Actor fills the description for the String Parameter as *Enter the subdomain name*
   Actor selects “String Parameter” under “Add parameter” dropdown
   Actor fills the name for the String Parameter as *subdomain_ip*
   Actor fills the description for the String Parameter as *Enter the subdomain IP*
   Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
   Actor selects “Git” from SCM dropdown
   Actor pastes the URL of the DevOps URL of the GitHub Repository
   Actor adds the credentials of the GitHub repository
   Actor adds the branch specifier as “*/development” under “Branches to build”
   Actor fills the script path of the new node Jenkins file which is available in GitHub repository
   Centos/wildcard_subdomain/Jenkinsfile
   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
   - Actor enters the credentials to login
   - Actor clicks “New Item” in the side menu
   - Actor fills the job name
   - Actor selects “Pipeline”
   - Actor clicks “Ok”
   - Actor selects “This Project is parameterized” under General
   - Actor selects “String Parameter” under “Add parameter” dropdown
   - Actor fills the name for the String Parameter as *subdomainName*
   - Actor fills the description for the String Parameter as *Enter the Frontend Subdomian*
   - Actor selects “String Parameter” under “Add parameter” dropdown
   - Actor fills the name for the String Parameter as *onboardingbackenduri*
   - Actor fills the description for the String Parameter as *Enter the Backend URL*
   - Actor selects “String Parameter” under “Add parameter” dropdown
   - Actor fills the name for the String Parameter as *keycloakuri*
   - Actor fills the description for the String Parameter as *Enter the Keycloak URL*
   - Actor selects “String Parameter” under “Add parameter” dropdown
   - Actor fills the name for the String Parameter as *billinguri*
   - Actor fills the description for the String Parameter as *BillingURL*
   - Actor selects “String Parameter” under “Add parameter” dropdown
   - Actor fills the name for the String Parameter as *jenkinsuri*
   - Actor fills the description for the String Parameter as *Enter the Jenkins URL*

Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *masterIP*
Actor fills the description for the String Parameter as *Master Ip address*
Actor selects “Password Parameter” under “Add parameter” dropdown
Actor fills the name for the Password Parameter as *masterPsw*
Actor fills the description for the Password Parameter as *Worker Ip address*
Actor selects “ String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *nodeIP*
Actor fills the description for the String Parameter as *Master passowrd*
Actor selects “ Password Parameter” under “Add parameter” dropdown
Actor fills the name for the Password Parameter as *nodePsw*
Actor fills the description for the Password Parameter as *worker Passowrd*
Actor selects “ Password Parameter” under “Add parameter” dropdown
Actor fills the name for the Password Parameter as *stripepublish_key*
Actor fills the description for the Password Parameter as *Enter the stripe_publish_key*
Actor selects “ Password Parameter” under “Add parameter” dropdown
Actor fills the name for the Password Parameter as *stripesecret_key*
Actor fills the description for the Password Parameter as *Enter the stripe_secret_key*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *stripetaxpercent*
Actor fills the description for the String Parameter as *Enter the stripe_tax_percent*
Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options
Actor selects “Git” from SCM dropdown
Actor pastes the URL of the DevOps URL of the GitHub Repository
Actor adds the credentials of the GitHub repository
Actor adds the branch specifier as “*/development” under “Branches to build”
Actor fills the script path of the Onboarding Jenkins file which is available in GitHub repository
Onboarding_Launch/Jenkinsfile

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

    Actor enters the credentials to login

    Actor clicks “New Item” in the side menu

    Actor fills the job name as ***“platformdeploy”***

    Actor selects “Pipeline”

    Actor clicks “Ok”

    Actor selects “This Project is parameterized” under General

    Actor selects “String Parameter” under “Add parameter” dropdown

    Actor fills the name for the String Parameter as *subdomainName*

    Actor fills the description for the String Parameter as *Enter the Frontend Subdomian*

    Actor selects “String Parameter” under “Add parameter” dropdown

    Actor fills the name for the String Parameter as *platfrombackenduri*

    Actor fills the description for the String Parameter as *Enter the Backend URL*

    Actor selects “String Parameter” under “Add parameter” dropdown

    Actor fills the name for the String Parameter as *keycloakuri*

    Actor fills the description for the String Parameter as *Enter the Keycloak URL*

    Actor selects “String Parameter” under “Add parameter” dropdown

    Actor fills the name for the String Parameter as *elasticfrontenduri*

    Actor fills the description for the String Parameter as *Enter the elasticfrontend URL*

    Actor selects “String Parameter” under “Add parameter” dropdown

    Actor fills the name for the String Parameter as *elasticbackenduri*

    Actor fills the description for the String Parameter as *Enter the elasticbackend URL*

    Actor selects “String Parameter” under “Add parameter” dropdown

    Actor fills the name for the String Parameter as *supporturi*

    Actor fills the description for the String Parameter as *Enter the support URL*

Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *imageuri*
Actor fills the description for the String Parameter as *Enter the image URL*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *miniouri*
Actor fills the description for the String Parameter as *Enter the minio URL*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *billinguri*
Actor fills the description for the String Parameter as *Enter the Billing URL*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *onboardingbackenduri*
Actor fills the description for the String Parameter as *Enter the Onboard Backend URL*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *userName*
Actor fills the description for the String Parameter as *Enter User name*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *email*
Actor fills the description for the String Parameter as *Enter user email*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *kUserID*
Actor fills the description for the String Parameter as *Enter KUID*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *stripepublish_key*
Actor fills the description for the String Parameter as *Enter the stripe_publish_key*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *stripetaxpercent*
Actor fills the description for the String Parameter as *Enter the stripe_tax_percent*
Actor selects “String Parameter” under “Add parameter” dropdown
Actor fills the name for the String Parameter as *adminemail*
Actor fills the Default Value for the String Parameter as *vignesh.k@akonisoftwareservices.com*

Actor fills the description for the String Parameter as *Admin url for ssl creation*

Actor selects “String Parameter” under “Add parameter” dropdown

Actor fills the name for the String Parameter as *masterIP*

Actor fills the Default Value for the String Parameter ***“CURRENT IP OF THE KUBERNETES MASTER”***

Actor fills the description for the String Parameter as *Master Ip address*

Actor selects “Password Parameter” under “Add parameter” dropdown

Actor fills the name for the Password Parameter as *masterPsw*

Actor fills the Default Value for the Password Parameter ***“CURRENT PASSWORD OF THE KUBERNETES MASTER”***

Actor fills the description for the Password Parameter as *Master password*

Actor selects “String Parameter” under “Add parameter” dropdown

Actor fills the name for the String Parameter as *nodeIP*

Actor fills the Default Value for the String Parameter ***“CURRENT IP OF THE PLATFORM NODE”***

Actor fills the description for the String Parameter as *Worker Ip address*

Actor selects “Password Parameter” under “Add parameter” dropdown

Actor fills the name for the Password Parameter as *nodePsw*

Actor fills the Default Value for the Password Parameter ***“CURRENT PASSWORD OF THE PLATFORM NODE”***

Actor fills the description for the Password Parameter as *worker Password*

Actor selects “Pipeline script from SCM” under Definition dropdown of Advanced Project options

Actor selects “Git” from SCM dropdown

Actor pastes the URL of the DevOps URL of the GitHub Repository

Actor adds the credentials of the GitHub repository

Actor adds the branch specifier as “*/development” under “Branches to build”

Actor fills the script path of the platform Jenkins file which is available in GitHub repository

Platform_Launch/Jenkinsfile

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
   Actor enters the credentials and logs in
   Actor clicks the Nexus Installation Jenkins Job
   Actor clicks “Build with Parameters” in the side menu
   Actor fills the Ip and password of the Nexus Droplet
   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

Actor sets the new password for the account

Actor clicks “Create Repository”

Actor fills the Name of the Repository

Actor checks and fills the HTTP port number as 8086

Actor checks the “Force Basic Authentication”

Actor checks the “Enable Docker V1 API” under “Docker Registry API Support”

Actor clicks “Save”

Actor clicks “Roles” under “Security” in the side menu

Actor clicks “Create Role”

Actor fills the Role ID

Actor fills the Role Description

Actor sets the privileges

nx-repository-admin-docker-CCS-edit

nx-repository-admin-docker-CCS-read

Actor clicks “Save”

Actor clicks “Users” under “Security” in the side menu

Actor clicks “Create local user”

Actor fills the ID

Actor fills the first name

Actor fills the last name

Actor fills the email address

Actor sets the status as “Active”

Actor sets the Role as “nx-admin”

Actor clicks “Save”

Actor clicks “Create local user” to create another user for Jenkins

Actor fills the ID as “Jenkins”

Actor fills the first name

Actor fills the last name
Actor fills the email address
Actor sets the status as “Disabled”
Actor sets the Role as “Jenkins”
Actor clicks “Save”
Actor clicks “Anonymous” under “Security” in the side menu
Actor unchecks “Allow Anonymous users to access the server”
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
   Actor enters the user credentials and logs in
   Actor clicks “Credentials” in the side menu
   Actor clicks “Jenkins” under “Stores scoped to Jenkins”
   Actor clicks “Global Credentials” under “System”
   Actor clicks “Adding some credentials” under “Global Credentials”
   Actor clicks “Add Credentials” in the side menu
   Actor fills the username of Nexus
   Actor fills the password of Nexus
   Actor fills the ID as “dockerhub”
   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

   Actor opens “daemon.json” file to edit

   vi daemon.json

   Actor updates the Nexus IP

   Actor saves the file and exit

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

   Actor opens “inventory” file to edit

   vi inventory

   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

   Actor saves the file and exit

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

   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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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”

    Actor updates the branch from Master to Development

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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

    Actor enters the credentials and logs in

    Actor clicks the Onboarding Front End Jenkins Job

    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

    Actor enters the credentials and logs in

    Actor clicks the Onboarding Back End Jenkins Job

    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

   Actor enters the credentials and logs in

   Actor clicks the Onboarding Key Cloak Jenkins Job

   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

    Actor enters the credentials and logs in

    Actor clicks the Onboarding Billing Jenkins Job

    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

    Actor enters the credentials and logs in

    Actor clicks the Platform Front End Jenkins Job

    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

    Actor enters the credentials and logs in

    Actor clicks the Platform Back End Jenkins Job

    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

   Actor enters the credentials and logs in

   Actor clicks the Platform Key Cloak Jenkins Job

   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

    Actor enters the credentials and logs in

    Actor clicks the Platform Elastic Search Jenkins Job

    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

    Actor enters the credentials and logs in

    Actor clicks the Platform Image Service Jenkins Job

    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:

Actor clicks “CCS-Onboarding-DevOps”

Actor updates the branch from Master to Development

Actor clicks “Centos”

Actor clicks “Ansible_Kubernetes”

Actor clicks “Jenkinsfile”

Actor clicks the Edit icon to edit it

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"

Actor enters the command and clicks ”Commit”

Actor clicks “Ansible_Kubernetes”

Actor clicks “daemon.json”

Actor clicks the Edit icon to edit it

Actor updates the Nexus IP

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
   Actor enters the credentials and logs in
   Actor clicks the Kubernetes Jenkins Job
   Actor clicks “Build with Parameters” in the side menu
   Actor fills the IP and password of the Master & Onboarding Node Droplet
   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”

    Actor updates the branch from Master to Development

    Actor clicks “Centos”

    Actor clicks “Adding_new_node”

    Actor clicks “Jenkinsfile”

    Actor clicks the Edit icon to edit it

    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"

    Actor enters the command and clicks ”Commit”

    Actor clicks “Adding_new_node”

    Actor clicks “daemon.json”

    Actor clicks the Edit icon to edit it

    Actor updates the Nexus IP

    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
    Actor enters the credentials and logs in
    Actor clicks the New_Node_Kubernetes Jenkins Job
    Actor clicks “Build with Parameters” in the side menu
    Actor fills the IP and password of the Master & Platform Node Droplet
    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

    Actor enters the credentials and logs in

    Actor clicks the SSL Jenkins Job

    Actor clicks “Build with Parameters” in the side menu

    Actor fills the IP and password of the Onboarding Node Droplet

    Actor choose the certificate name under “folder_name” dropdown as per their expectations

    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
    Actor enters the credentials and logs in
    Actor clicks the SSL Jenkins Job
    Actor clicks “Build with Parameters” in the side menu
    Actor fills the IP and password of the Platform Node Droplet
    Actor choose the certificate name under “folder_name” dropdown as per their expectations
    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

    Actor enters the credentials and logs in

    Actor clicks the DNS Jenkins Job

    Actor clicks “Build with Parameters” in the side menu

    Actor fills the domain name

    Actor fills the Subdomain name with Onboarding Front End domain

    Actor fills the IP of the Onboarding Node Droplet

    Actor clicks “Build”

    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
    Actor enters the credentials and logs in
    Actor clicks the DNS Jenkins Job
    Actor clicks “Build with Parameters” in the side menu
    Actor fills the domain name
    Actor fills the Subdomain name with “*”
    Actor fills the IP of the Platform Node Droplet
    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”

Actor updates the branch from Master to Development

Actor clicks “Onboarding_Launch”

Actor clicks “onboarding_keycloak_fullstack”

Actor clicks “App_stack_ansible.yaml”

Actor clicks the Edit icon to edit it

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 }}"

Actor enters the command and clicks ”Commit”

Actor redirects to “onboarding_keycloak_fullstack” by clicking it

Actor clicks “Keycloak”

Actor clicks “keycloak_deploy.yaml”

Actor clicks the Edit icon to edit it

Actor updates the Nexus credentials in the command under “Containers”

- image: <Nexus Droplet IP>:8086/onboard_keycloak:<Version tag from the Nexus UI>

Actor enters the command and clicks ”Commit”

Actor redirects to “onboarding_keycloak_fullstack” by clicking it

Actor clicks “Onboarding”

Actor clicks “onboarding_app”

Actor clicks “backend_deployment.yaml”

Actor clicks the Edit icon to edit it

Actor updates the Nexus credentials in the command under “Containers”

- image: <Nexus Droplet IP>:8086/onboard_backend:<Version tag from the Nexus UI>

Actor enters the command and clicks ”Commit”

Actor redirects to “onboarding_app” by clicking it

Actor clicks “frontent_deploy.yaml”

Actor clicks the Edit icon to edit it

Actor updates the Nexus credentials in the command under “Containers”

- image: <Nexus Droplet IP>:8086/onboard_frontend:<Version tag from the Nexus UI>

Actor enters the command and clicks ”Commit”

Actor redirects to “Onboarding” by clicking it

Actor clicks “onboarding_billing_app”

Actor clicks “billing_deploy.yaml”

Actor clicks the Edit icon to edit it

Actor updates the Nexus credentials in the command under “Containers”

- image: <Nexus Droplet IP>:8086/onboard_billing:<Version tag from the Nexus UI>

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”
1. Actor updates the branch from Master to Development
2. Actor clicks “Platform_Launch”
3. Actor clicks “Jenkinsfile”
4. Actor clicks the Edit icon to edit it
5. 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")
6. Actor enters the command and clicks ”Commit”
7. Actor redirects to “Platform_Launch” by clicking it
8. Actor clicks “platform_keycloak_fullstack”
9. Actor clicks “App_stack_ansible.yaml”
10. Actor clicks the Edit icon to edit it
11. 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 }}"
12. Actor enters the command and clicks ”Commit”
13. Actor redirects to “platform_keycloak_fullstack” by clicking it
14. Actor clicks “Keycloak”

15. Actor clicks “keycloak_deploy.yaml”
16. Actor clicks the Edit icon to edit it
17. Actor updates the Nexus credentials in the command under “Containers”
    - image: <Nexus Droplet IP>:8086/platform_keycloak:<Version tag from the Nexus UI>
18. Actor enters the command and clicks ”Commit”
19. Actor redirects to “platform_keycloak_fullstack” by clicking it
20. Actor clicks “Platform”
21. Actor clicks “platform_elasticsearch”
22. Actor clicks “elasticbackend_deployment.yaml”
23. Actor clicks the Edit icon to edit it
24. Actor updates the Nexus credentials in the command under “Containers”
    - image: <Nexus Droplet IP>:8086/platform_elasticsearch:<Version tag from the Nexus UI>
25. Actor enters the command and clicks ”Commit”
26. Actor redirects to “Platform” by clicking it
27. Actor clicks “platform_app”
28. Actor clicks “backend_deployment.yaml”
29. Actor clicks the Edit icon to edit it
30. Actor updates the Nexus credentials in the command under “Containers”
    - image: <Nexus Droplet IP>:8086/platform_backend:<Version tag from the Nexus UI>
31. Actor enters the command and clicks ”Commit”
32. Actor redirects to “Platform” by clicking it
33. Actor clicks “platform_app”
34. Actor clicks “frontend_deploy.yaml”
35. Actor clicks the Edit icon to edit it
36. Actor updates the Nexus credentials in the command under “Containers”
    - image: <Nexus Droplet IP>:8086/platform_frontend:<Version tag from the Nexus UI>
37. Actor enters the command and clicks ”Commit”
38. Actor redirects to “Platform” by clicking it
39. Actor clicks “platform_imageserver”
40. Actor clicks “imagebackend_deployment.yaml”

41.Actor clicks the Edit icon to edit it

42.Actor updates the Nexus credentials in the command under “Containers”

- image: <Nexus Droplet IP>:8086/platform_image:<Version tag from the Nexus UI>

43.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

    Actor enters the credentials and logs in

    Actor clicks the Onboarding Deploy Jenkins Job

    Actor clicks “Build with Parameters” in the side menu

    Actor fills the Onboarding frontend url in without “https” and remaining url’s with “https” in parameters fields.

    Actor clicks “Build”

Alternate Workflow: None
Post – Condition: Onboarding successfully deployed to Master and Onboarding Node Droplet