**README.md**

**Project 501: Microservices CI/CD Pipeline**

**Description**

This project aims to create full CI/CD Pipeline for microservice based applications using [Spring Petclinic Microservices Application](https://github.com/spring-petclinic/spring-petclinic-microservices). Jenkins Server deployed on Elastic Compute Cloud (EC2) Instance is used as CI/CD Server to build pipelines.

**Introduction**

Click to see project planning cycles of [the Petclinic Application](https://github.com/clarusway/clarusway-aws-devops-pro-3-20/blob/master/devops/projects/501-microservices-ci-cd-pipeline/microservice-project-501-planning.pdf).

**Flow of Tasks for Project Realization**

| **Epic** | **Task** | **Students Task** | **Task Definition** | **Branch** |
| --- | --- | --- | --- | --- |
| Local Development Environment | Prepare Development Server Manually on EC2 Instance | MSP-1 | Prepare development server manually on Amazon Linux 2 for developers, enabled with Docker, Docker-Compose, Java 11, Git. |  |
| Local Development Environment | Prepare GitHub Repository for the Project | MSP-2-1 | Fork and clone the Petclinic app from the Clarusway repository [Petclinic Microservices Application](https://github.com/clarusway/petclinic-microservices.git) |  |
| Local Development Environment | Prepare GitHub Repository for the Project | MSP-2-2 | Prepare base branches namely master, dev, release for DevOps cycle. |  |
| Local Development Environment | Check the Maven Build Setup on Dev Branch | MSP-3 | Check the Maven builds for test, package, and install phases on dev branch |  |
| Local Development Environment | Prepare a Script for Packaging the Application | MSP-4 | Prepare a script to package the application with Maven wrapper | feature/msp-4 |
| Local Development Environment | Prepare Development Server Cloudformation Template | MSP-5 | Prepare development server script with Cloudformation template for developers, enabled with Docker, Docker-Compose, Java 11, Git. | feature/msp-5 |
| Local Development Build | Prepare Dockerfiles for Microservices | MSP-6 | Prepare Dockerfiles for each microservices. | feature/msp-6 |
| Local Development Build | Prepare Script for Building Docker Images | MSP-7 | Prepare a script to package and build the docker images for all microservices. | feature/msp-7 |
| Local Development Build | Create Docker Compose File for Local Development | MSP-8-1 | Prepare docker compose file to deploy the application locally. | feature/msp-8 |
| Local Development Build | Create Docker Compose File for Local Development | MSP-8-2 | Prepare a script to test the deployment of the app locally. | feature/msp-8 |
| Testing Environment Setup | Implement Unit Tests | MSP-9-1 | Implement 3 Unit Tests locally. | feature/msp-9 |
| Testing Environment Setup | Setup Code Coverage Tool | MSP-9-2 | Update POM file for Code Coverage Report. | feature/msp-9 |
| Testing Environment Setup | Implement Code Coverage | MSP-9-3 | Generate Code Coverage Report manually. | feature/msp-9 |
| Testing Environment Setup | Prepare Selenium Tests | MSP-10-1 | Prepare 3 Selenium Jobs for QA Automation Tests. | feature/msp-10 |
| Testing Environment Setup | Implement Selenium Tests | MSP-10-2 | Run 3 Selenium Tests against local environment. | feature/msp-10 |
| CI Server Setup | Prepare Jenkins Server | MSP-11 | Prepare Jenkins Server for CI/CD Pipeline. | feature/msp-11 |
| CI Server Setup | Configure Jenkins Server for Project | MSP-12 | Configure Jenkins Server for Project Setup. |  |
| CI Server Setup | Prepare CI Pipeline | MSP-13 | Prepare CI pipeline (UT only) for all dev, feature and bugfix branches. | feature/msp-13 |
| Registry Setup for Development | Create Docker Registry for Dev Manually | MSP-14 | Create Docker Registry on AWS ECR manually using Jenkins job. |  |
| Registry Setup for Development | Prepare Script for Docker Registry | MSP-15 | Prepare a script to create Docker Registry on AWS ECR using Jenkins job. | feature/msp-15 |
| QA Automation Setup for Development | Create a QA Automation Environment | MSP-16 | Create a QA Automation Environment with Docker Swarm. | feature/msp-16 |
| QA Automation Setup for Development | Prepare a QA Automation Pipeline | MSP-17 | Prepare a QA Automation Pipeline on dev branch for Nightly Builds. | feature/msp-17 |
| QA Setup for Release | Create a Key Pair for QA Environment | MSP-18-1 | Create a permanent Key Pair for Ansible to be used in QA Environment on Release. | feature/msp-18 |
| QA Setup for Release | Create a QA Infrastructure with AWS Cloudformation | MSP-18-2 | Create a Permanent QA Infrastructure for Docker Swarm with AWS Cloudformation. | feature/msp-18 |
| QA Setup for Release | Create a QA Environment with Docker Swarm | MSP-18-3 | Create a QA Environment for Release with Docker Swarm. | feature/msp-18 |
| QA Setup for Release | Prepare Build Scripts for QA Environment | MSP-19 | Prepare Build Scripts for creating ECR Repo for QA, building QA Docker images, deploying app with Docker Compose. | feature/msp-19 |
| QA Setup for Release | Build and Deploy App on QA Environment Manually | MSP-20 | Build and Deploy App for QA Environment Manually using Jenkins Jobs. | feature/msp-20 |
| QA Setup for Release | Prepare a QA Pipeline | MSP-21 | Prepare a QA Pipeline using Jenkins on release branch for Weekly Builds. | feature/msp-21 |

**MSP 1 - Prepare Development Server Manually on EC2 Instance**

* Prepare development server manually on Amazon Linux 2 for developers, enabled with Docker, Docker-Compose, Java 11, Git.

# update os

sudo yum update -y

# set hostname as petclinic-dev-server

sudo hostnamectl set-hostname petclinic-dev-server

# install docker

sudo amazon-linux-extras install docker -y

sudo systemctl start docker

sudo systemctl enable docker

sudo usermod -a -G docker ec2-user

# install docker compose

sudo curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

# install JDK 11

sudo yum install java-11-amazon-corretto -y

# install Git

sudo yum install git -y

**MSP 2 - Prepare GitHub Repository for the Project**

* Fork the Petclinic app from the Clarusway repository [Petclinic Microservices Application](https://github.com/clarusway/petclinic-microservices.git)
* Rename the forked repo on your GitHub as microservices-ci-cd-pipeline-with-petclinic-app.
* Clone the forked repo from your GitHub repo on development server.

git clone <your-repo-url>

* Prepare base branches namely master, dev, release for DevOps cycle.
  + Create dev base branch.
  + git branch # show local branches
  + git branch -a # shows all branches local and remote
  + git checkout master
  + git branch dev
  + git checkout dev

git push --set-upstream origin dev

* + Create release base branch.
  + git fetch # will get latest info from the remote repo
  + git checkout master
  + git branch release
  + git checkout release

git push --set-upstream origin release

**MSP 3 - Check the Maven Build Setup on Dev Branch**

* Switch to dev branch.

git checkout dev

* Test the compiled source code.

./mvnw clean test

* Take the compiled code and package it in its distributable JAR format.

./mvnw package -Dmaven.test.skip=true -Dmaven.compile.skip=true

* Install distributable JARs into local repository.

./mvn clean install

**MSP 4 - Prepare a Script for Packaging the Application**

* Create feature/msp-4 branch from dev.

git checkout dev

git branch feature/msp-4

git checkout feature/msp-4

* Prepare a script to package the application with maven wrapper and save it as package-with-mvn-wrapper.sh.

./mvnw clean package

* Commit and push the new script to remote repo.

git add .

git commit -m "added packaging script"

git push --set-upstream origin feature/msp-4

git checkout dev

git merge feature/msp-4

git push origin dev

**MSP 5 - Prepare Development Server Cloudformation Template**

* Create feature/msp-5 branch from dev.

git checkout dev

git checkout -b feature/msp-5

* Create a folder for infrastructure setup with the name of infrastructure.

mkdir infrastructure

* Prepare development server script with Cloudformation template for developers, enabled with Docker, Docker-Compose, Java 11, Git and save it as dev-server-for-petclinic-app-cfn-template.yml under infrastructure folder.

# update os

sudo yum update -y

# set hostname as petclinic-dev-server

sudo hostnamectl set-hostname petclinic-dev-server

# install docker

sudo amazon-linux-extras install docker -y

sudo systemctl start docker

sudo systemctl enable docker

sudo usermod -a -G docker ec2-user

# install docker compose

sudo curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

# install JDK 11

sudo yum install java-11-amazon-corretto -y

# install Git

sudo yum install git -y

* Commit and push the new script to remote repo.

git add .

git commit -m "added cloudformation template for dev server"

git push --set-upstream origin feature/msp-5

git checkout dev

git merge feature/msp-5

git push origin dev

**MSP 6 - Prepare Dockerfiles for Microservices**

* Create feature/msp-6 branch from dev.

git checkout dev

git checkout -b feature/msp-6

* Prepare a Dockerfile for the admin-server microservice with following content and save it under spring-petclinic-admin-server.

FROM openjdk:11-jre

ARG DOCKERIZE\_VERSION=v0.6.1

ARG EXPOSED\_PORT=9090

ENV SPRING\_PROFILES\_ACTIVE docker

ADD https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE\_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE\_VERSION}.tar.gz dockerize.tar.gz

RUN tar xzf dockerize.tar.gz

RUN chmod +x dockerize

ADD ./target/\*.jar /app.jar

EXPOSE ${EXPOSED\_PORT}

ENTRYPOINT [ "java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

* Prepare a Dockerfile for the api-gateway microservice with the following content and save it under spring-petclinic-api-gateway.

FROM openjdk:11-jre

ARG DOCKERIZE\_VERSION=v0.6.1

ARG EXPOSED\_PORT=8080

ENV SPRING\_PROFILES\_ACTIVE docker

ADD https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE\_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE\_VERSION}.tar.gz dockerize.tar.gz

RUN tar xzf dockerize.tar.gz

RUN chmod +x dockerize

ADD ./target/\*.jar /app.jar

EXPOSE ${EXPOSED\_PORT}

ENTRYPOINT [ "java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

* Prepare a Dockerfile for the config-server microservice with the following content and save it under spring-petclinic-config-server.

FROM openjdk:11-jre

ARG DOCKERIZE\_VERSION=v0.6.1

ARG EXPOSED\_PORT=8888

ENV SPRING\_PROFILES\_ACTIVE docker

ADD https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE\_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE\_VERSION}.tar.gz dockerize.tar.gz

RUN tar xzf dockerize.tar.gz

RUN chmod +x dockerize

ADD ./target/\*.jar /app.jar

EXPOSE ${EXPOSED\_PORT}

ENTRYPOINT [ "java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

* Prepare a Dockerfile for the customer-service microservice with the following content and save it under spring-petclinic-customer-service.

FROM openjdk:11-jre

ARG DOCKERIZE\_VERSION=v0.6.1

ARG EXPOSED\_PORT=8081

ENV SPRING\_PROFILES\_ACTIVE docker

ADD https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE\_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE\_VERSION}.tar.gz dockerize.tar.gz

RUN tar xzf dockerize.tar.gz

RUN chmod +x dockerize

ADD ./target/\*.jar /app.jar

EXPOSE ${EXPOSED\_PORT}

ENTRYPOINT [ "java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

* Prepare a Dockerfile for the discovery-server microservice with the following content and save it under spring-petclinic-discovery-server.

FROM openjdk:11-jre

ARG DOCKERIZE\_VERSION=v0.6.1

ARG EXPOSED\_PORT=8761

ENV SPRING\_PROFILES\_ACTIVE docker

ADD https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE\_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE\_VERSION}.tar.gz dockerize.tar.gz

RUN tar xzf dockerize.tar.gz

RUN chmod +x dockerize

ADD ./target/\*.jar /app.jar

EXPOSE ${EXPOSED\_PORT}

ENTRYPOINT [ "java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

* Prepare a Dockerfile for the hystrix-dashboard microservice with the following content and save it under spring-petclinic-hystrix-dashboard.

FROM openjdk:11-jre

ARG DOCKERIZE\_VERSION=v0.6.1

ARG EXPOSED\_PORT=7979

ENV SPRING\_PROFILES\_ACTIVE docker

ADD https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE\_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE\_VERSION}.tar.gz dockerize.tar.gz

RUN tar xzf dockerize.tar.gz

RUN chmod +x dockerize

ADD ./target/\*.jar /app.jar

EXPOSE ${EXPOSED\_PORT}

ENTRYPOINT [ "java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

* Prepare a Dockerfile for the vets-service microservice with the following content and save it under spring-petclinic-vets-service.

FROM openjdk:11-jre

ARG DOCKERIZE\_VERSION=v0.6.1

ARG EXPOSED\_PORT=8083

ENV SPRING\_PROFILES\_ACTIVE docker

ADD https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE\_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE\_VERSION}.tar.gz dockerize.tar.gz

RUN tar xzf dockerize.tar.gz

RUN chmod +x dockerize

ADD ./target/\*.jar /app.jar

EXPOSE ${EXPOSED\_PORT}

ENTRYPOINT [ "java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

* Prepare a Dockerfile for the visits-service microservice with the following content and save it under spring-petclinic-visits-service.

FROM openjdk:11-jre

ARG DOCKERIZE\_VERSION=v0.6.1

ARG EXPOSED\_PORT=8082

ENV SPRING\_PROFILES\_ACTIVE docker

ADD https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE\_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE\_VERSION}.tar.gz dockerize.tar.gz

RUN tar xzf dockerize.tar.gz

RUN chmod +x dockerize

ADD ./target/\*.jar /app.jar

EXPOSE ${EXPOSED\_PORT}

ENTRYPOINT [ "java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

* Commit the changes, then push the Dockerfiles to the remote repo.

git add .

git commit -m "added Dockerfiles for microservices"

git push --set-upstream origin feature/msp-6

git checkout dev

git merge feature/msp-6

git push origin dev

**MSP 7 - Prepare Script for Building Docker Images**

* Create feature/msp-7 branch from dev.

git checkout dev

git checkout -b feature/msp-7

* Prepare a script to build the docker images and save it as build-dev-docker-images.sh.

./package-with-mvn-wrapper.sh

docker build --force-rm -t "petclinic-admin-server:dev" ./spring-petclinic-admin-server

docker build --force-rm -t "petclinic-api-gateway:dev" ./spring-petclinic-api-gateway

docker build --force-rm -t "petclinic-config-server:dev" ./spring-petclinic-config-server

docker build --force-rm -t "petclinic-customers-service:dev" ./spring-petclinic-customers-service

docker build --force-rm -t "petclinic-discovery-server:dev" ./spring-petclinic-discovery-server

docker build --force-rm -t "petclinic-hystrix-dashboard:dev" ./spring-petclinic-hystrix-dashboard

docker build --force-rm -t "petclinic-vets-service:dev" ./spring-petclinic-vets-service

docker build --force-rm -t "petclinic-visits-service:dev" ./spring-petclinic-visits-service

docker build --force-rm -t "petclinic-prometheus-server:dev" ./docker/prometheus

docker build --force-rm -t "petclinic-grafana-server:dev" ./docker/grafana

* Commit the changes, then push the new script to the remote repo.

git add .

git commit -m "added build script "

git push --set-upstream origin feature/msp-7

git checkout dev

git merge feature/msp-7

git push origin dev

**MSP 8 - Create Docker Compose File for Local Development**

* Create feature/msp-8 branch from dev.

git checkout dev

git checkout -b feature/msp-8

* Prepare docker compose file to deploy the application locally and save it as docker-compose-local.yml.

version: '2'

services:

config-server:

image: petclinic-config-server:dev

container\_name: config-server

mem\_limit: 512M

ports:

- 8888:8888

discovery-server:

image: petclinic-discovery-server:dev

container\_name: discovery-server

mem\_limit: 512M

ports:

- 8761:8761

depends\_on:

- config-server

entrypoint: ["./dockerize", "-wait=tcp://config-server:8888", "-timeout=60s", "--", "java", "-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]

customers-service:

image: petclinic-customers-service:dev

container\_name: customers-service

mem\_limit: 512M

ports:

- 8081:8081

depends\_on:

- config-server

- discovery-server

entrypoint: ["./dockerize", "-wait=tcp://discovery-server:8761", "-timeout=60s", "--", "java", "-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

visits-service:

image: petclinic-visits-service:dev

container\_name: visits-service

mem\_limit: 512M

ports:

- 8082:8082

depends\_on:

- config-server

- discovery-server

entrypoint: ["./dockerize", "-wait=tcp://discovery-server:8761", "-timeout=60s", "--", "java", "-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

vets-service:

image: petclinic-vets-service:dev

container\_name: vets-service

mem\_limit: 512M

ports:

- 8083:8083

depends\_on:

- config-server

- discovery-server

entrypoint: ["./dockerize", "-wait=tcp://discovery-server:8761", "-timeout=60s", "--", "java", "-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

api-gateway:

image: petclinic-api-gateway:dev

container\_name: api-gateway

mem\_limit: 512M

ports:

- 8080:8080

depends\_on:

- config-server

- discovery-server

entrypoint: ["./dockerize", "-wait=tcp://discovery-server:8761", "-timeout=60s", "--", "java", "-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

admin-server:

image: petclinic-admin-server:dev

container\_name: admin-server

mem\_limit: 512M

ports:

- 9090:9090

depends\_on:

- config-server

- discovery-server

entrypoint: ["./dockerize", "-wait=tcp://discovery-server:8761", "-timeout=60s", "--", "java", "-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

hystrix-dashboard:

image: petclinic-hystrix-dashboard:dev

container\_name: hystrix-dashboard

mem\_limit: 512M

ports:

- 7979:7979

depends\_on:

- config-server

- discovery-server

entrypoint: ["./dockerize", "-wait=tcp://discovery-server:8761", "-timeout=60s", "--", "java", "-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar" ]

tracing-server:

image: openzipkin/zipkin

container\_name: tracing-server

mem\_limit: 512M

environment:

- JAVA\_OPTS=-XX:+UnlockExperimentalVMOptions -Djava.security.egd=file:/dev/./urandom

ports:

- 9411:9411

grafana-server:

image: petclinic-grafana-server:dev

container\_name: grafana-server

mem\_limit: 256M

ports:

- 3000:3000

prometheus-server:

image: petclinic-prometheus-server:dev

container\_name: prometheus-server

mem\_limit: 256M

ports:

- 9091:9090

* Prepare a script to test the deployment of the app locally with docker-compose-local.yml and save it as test-local-deployment.sh.
* Commit the change, then push the docker compose file to the remote repo.

**MSP 9 - Setup Unit Tests and Configure Code Coverage Report**

* Create feature/msp-9 branch from dev.
* Create following unit tests for Pet.java under customer-service microservice using the following PetTest class and save it as PetTest.java under ./spring-petclinic-customers-service/src/test/java/org/springframework/samples/petclinic/customers/model/ folder.
* Commit the change, then push the changes to the remote repo.
* Implement unit tests with maven wrapper for only customer-service microservice locally on Dev Server.
* Update POM file at root folder for Code Coverage Report using Jacoco tool plugin.
* Commit the change, then push the changes to the remote repo.
* Create code coverage report for only customer-service microservice locally on Dev Server.
* Deploy code coverage report (located under relative path target/site/jacoco of the microservice) on Simple HTTP Server for only customer-service microservice on Dev Server.

**MSP 10 - Prepare and Implement Selenium Tests**

* Create feature/msp-10 branch from dev.
* Create a folder for Selenium jobs with the name of selenium-jobs.
* Create Selenium job (QA Automation test) for testing Owners >> All page and save it as test\_owners\_all\_headless.py under selenium-jobs folder.
* Create Selenium job (QA Automation test) for testing Owners >> Register page and save it as test\_owners\_register\_headless.py under selenium-jobs folder.
* Create Selenium job (QA Automation test) for testing Veterinarians page and save it as test\_veterinarians\_headless.py under selenium-jobs folder.
* Commit the change, then push the selenium jobs to the remote repo.

**MSP 11 - Prepare Jenkins Server for CI/CD Pipeline**

* Create feature/msp-11 branch from dev.
* Set up a Jenkins Server and enable it with Git, Docker, Docker Compose, AWS CLI v2, ECR Credential Helper, Python3, Ansible and Boto3. To do so, prepare a [Cloudformation template for Jenkins Server](https://github.com/clarusway/clarusway-aws-devops-pro-3-20/blob/master/devops/projects/501-microservices-ci-cd-pipeline/msp-11-jenkins-server-cfn-template.yml) with following script and save it as jenkins-server-cfn-template.yml under infrastructure folder.
* Grant permissions to Jenkins Server within Cloudformation template to create Cloudformation Stack and create ECR Registry, push or pull Docker images to ECR Repo.
* Commit the change, then push the cloudformation file to the remote repo.

**MSP 12 - Configure Jenkins Server for Project**

* Get the initial administrative password.
* Enter the temporary password to unlock the Jenkins.
* Install suggested plugins.
* Create first admin user.
* Open your Jenkins dashboard and navigate to Manage Jenkins >> Manage Plugins >> Available tab
* Search and select GitHub Integration, Docker Plugin, Docker Pipeline, and Jacoco plugins, then click Install without restart. Note: No need to install the other Git plugin which is already installed can be seen under Installed tab.
* Configure Docker as cloud agent by navigating to Manage Jenkins >> Manage Nodes and Clouds >> Configure Clouds and using tcp://localhost:2375 as Docker Host URI.

**MSP 13 - Prepare Continuous Integration (CI) Pipeline**

* Create feature/msp-13 branch from dev.
* Create a folder, named jenkins, to keep Jenkinsfiles and Jenkins jobs of the project.
* Create a Jenkins job with name of petclinic-ci-job with following script to unit tests and configure a webhook to trigger the job. Jenkins CI Job should be triggered to run on each commit of feature\*\* and bugfix\*\* branches and on each PR merge to dev branch.
* Prepare a script for Jenkins CI job (covering Unit Test only) and save it as jenkins-petclinic-ci-job.sh under jenkins folder.
* Add post-build action to Jenkins Job to record Jacoco Coverage Report.
* Create a webhook for Jenkins CI Job;
  + Go to the project repository page and click on Settings.
  + Click on the Webhooks on the left hand menu, and then click on Add webhook.
  + Copy the Jenkins URL, paste it into Payload URL field, add /github-webhook/ at the end of URL, and click on Add webhook.
* http://[jenkins-server-hostname]:8080/github-webhook/
* Commit the change, then push the Jenkinsfile to the remote repo.

**MSP 14 - Create Docker Registry for Dev Manually**

* Create a Jenkins Job and name it as create-ecr-docker-registry-for-dev to create Docker Registry for dev on AWS ECR manually.

**MSP 15 - Prepare Script for Development Docker Registry**

* Create feature/msp-15 branch from dev.
* Prepare a script to create Docker Registry for dev on AWS ECR and save it as create-ecr-docker-registry-for-dev.sh under infrastructure folder.
* Commit the change, then push the script to the remote repo.

**MSP 16 - Create a QA Automation Environment with Docker Swarm**

* Create feature/msp-16 branch from dev.
* Prepare a Cloudformation template for Docker Swarm Infrastructure consisting of 3 Managers, 2 Worker Instances and save it as docker-swarm-infrastructure-cfn-template.yml under infrastructure folder.
* Grant permissions to Docker Machines within Cloudformation template to create ECR Registry, push or pull Docker images to/from ECR Repo.
* Commit the change, then push the cloudformation template to the remote repo.
* Create a Jenkins Job and name it as test-creating-qa-automation-infrastructure to test bash scripts creating QA Automation Infrastructure for dev manually.
* Check the environment tools setup and versions with following script.
* Test creating key pair for ansible using AWS CLI with following script.
* Test creating Docker Swarm infrastructure with AWS Cloudformation using AWS CLI with following script.
* Test SSH connection with one of the docker instance.
* Prepare static inventory file with name of hosts.ini for Ansible under ansible/inventory folder using Docker machines private IP addresses.
* Commit the change, then push the cloudformation template to the remote repo.
* Test ansible by pinging static hosts.
* Prepare dynamic inventory file with name of dev\_stack\_dynamic\_inventory\_aws\_ec2.yaml for Ansible under ansible/inventory folder using Docker machines private IP addresses.
* Prepare dynamic inventory file with name of dev\_stack\_swarm\_grand\_master\_aws\_ec2.yaml for Ansible under ansible/inventory folder using Docker machines private IP addresses.
* Prepare dynamic inventory file with name of dev\_stack\_swarm\_managers\_aws\_ec2.yaml for Ansible under ansible/inventory folder using Docker machines private IP addresses.
* Prepare dynamic inventory file with name of dev\_stack\_swarm\_workers\_aws\_ec2.yaml for Ansible under ansible/inventory folder using Docker machines private IP addresses.
* Commit the change, then push the cloudformation template to the remote repo.
* Check the Ansible dynamic inventory for dev environment.
* Test all instances within dev dynamic inventory by pinging static hosts.
* Create a ansible playbook to install and configure tools (Docker, Docker-Compose, AWS CLI V2, ECR Credential Helper) needed for all Docker Swarm nodes (instances) and save it as pb\_setup\_for\_all\_docker\_swarm\_instances.yaml under ansible/playbooks folder.
* Create a ansible playbook to initialize the Docker Swarm and configure tools on Grand Master instance of Docker Swarm and save it as pb\_initialize\_docker\_swarm.yaml under ansible/playbooks folder.
* Create a ansible playbook to join the Docker manager nodes to the Swarm and save it as pb\_join\_docker\_swarm\_managers.yaml under ansible/playbooks folder.
* Create a ansible playbook to join the Docker worker nodes to the Swarm and save it as pb\_join\_docker\_swarm\_workers.yaml under ansible/playbooks folder.
* Commit the change, then push the ansible playbooks to the remote repo.
* Test the playbooks to create a Docker Swarm on Cloudformation Stack.
* Test tearing down the Docker Swarm infrastructure using AWS CLI with following script.
* Test deleting existing key pair using AWS CLI with following script.
* Create a script to create QA Automation infrastructure and save it as create-qa-automation-environment.sh under infrastructure folder.
* Commit the change, then push the script to the remote repo.

**MSP 17 - Prepare a QA Automation Pipeline for Nightly Builds**

* Create feature/msp-17 branch from dev.
* Prepare a script to create ECR tags for the dev docker images and save it as package-with-maven-container.sh and save it under jenkins folder.
* Prepare a script to create ECR tags for the dev docker images and save it as prepare-tags-ecr-for-dev-docker-images.sh and save it under jenkins folder.
* Prepare a script to build the dev docker images tagged for ECR registry and save it as build-dev-docker-images-for-ecr.sh and save it under jenkins folder.
* Prepare a script to push the dev docker images to the ECR repo and save it as push-dev-docker-images-to-ecr.sh and save it under jenkins folder.
* Commit the change, then push the scripts to the remote repo.
* Prepare a docker compose file for swarm deployment and save it as docker-compose-swarm-dev.yml.
* Create Ansible playbook for deploying app on Docker swarm using docker compose file and save it as pb\_deploy\_app\_on\_docker\_swarm.yaml under ansible/playbooks folder.
* Prepare a script to deploy the application on docker swarm and save it as deploy\_app\_on\_docker\_swarm.sh under ansible/scripts folder.
* Create Selenium dummy test with name of dummy\_selenium\_test\_headless.py with following content to check the setup for the Selenium jobs and save it under selenium-jobs folder.
* Create Ansible playbook for running dummy selenium job and save it as pb\_run\_dummy\_selenium\_job.yaml under ansible/playbooks folder.
* Prepare a script to run the playbook for dummy selenium job on Jenkins Server (localhost) and save it as run\_dummy\_selenium\_job.sh under ansible/scripts folder.
* Commit the change, then push the scripts for dummy selenium job to the remote repo.
* Create a Jenkins job with name of test-running-dummy-selenium-job to check the setup for selenium tests by running dummy selenium job on feature/msp-17 branch.
* Create Ansible playbook for running all selenium jobs under ``selenium-jobsfolder and save it aspb\_run\_selenium\_jobs.yaml` under `ansible/playbooks` folder.
* Prepare a script to run the playbook for all selenium jobs on Jenkins Server (localhost) and save it as run\_selenium\_jobs.sh under ansible/scripts folder.
* Update the selenium jobs to get Docker Grand Master Pubic IP address as environment variable.
* Create a Jenkins pipeline with name of petclinic-nightly with following script to run QA automation tests and configure a cron job to trigger the pipeline every night at midnight (0 0 \* \* \*) on dev branch. Petclinic nightly build pipeline should be built on temporary QA automation environment.
* Prepare a Jenkinsfile for petclinic-nightly builds and save it as jenkinsfile-petclinic-nightly under jenkins folder.
* Commit the change, then push the script to the remote repo.

**MSP 18 - Create a QA Environment on Docker Swarm with Clouldformation and Ansible**

* Create feature/msp-18 branch from dev.
* Prepare a Cloudformation template for QA Docker Swarm Infrastructure consisting of 3 Managers, 2 Worker Instances and save it as qa-docker-swarm-infrastructure-cfn-template.yml under infrastructure folder.
* Grant permissions to Docker Machines within Cloudformation template to create ECR Registry, push or pull Docker images to/from ECR Repo.
* Create a Jenkins Job with the name of create-permanent-key-pair-for-petclinic-qa-env for Ansible key pair to be used in QA environment using following script, and save the script as create-permanent-key-pair-for-qa-environment.sh under jenkins folder.
* Create a Permanent QA Infrastructure for Docker Swarm with AWS Cloudformation using AWS CLI with following script.
* Prepare dynamic inventory file with name of qa\_stack\_dynamic\_inventory\_aws\_ec2.yaml for Ansible under ansible/inventory folder using Docker machines private IP addresses.
* Prepare script to create a QA Environment for Release on Docker Swarm using the same playbooks created for Dev environment.
* Prepare a Jenkinsfile to create a QA Environment on Docker Swarm manually and save it as jenkinsfile-create-qa-environment-on-docker-swarm under jenkins folder.
* Create a pipeline on Jenkins Server with name of create-qa-environment-on-docker-swarm and create QA environment manually on dev branch.
* Commit the change, then push the scripts to the remote repo.

**MSP 19 - Prepare Build Scripts for QA Environment**

* Create feature/msp-19 branch from dev.
* Create a Jenkins Job and name it as create-ecr-docker-registry-for-petclinic-qa to create Docker Registry for QA manually on AWS ECR.
* Prepare a script to create ECR tags for the dev docker images and save it as prepare-tags-ecr-for-qa-docker-images.sh and save it under jenkins folder.
* Prepare a script to build the dev docker images tagged for ECR registry and save it as build-qa-docker-images-for-ecr.sh and save it under jenkins folder.
* Prepare a script to push the dev docker images to the ECR repo and save it as push-qa-docker-images-to-ecr.sh and save it under jenkins folder.
* Prepare a docker compose file for swarm deployment on QA environment and save it as docker-compose-swarm-qa.yml.
* Create Ansible playbook for deploying app on QA environment using docker compose file and save it as pb\_deploy\_app\_on\_qa\_environment.yaml under ansible/playbooks folder.
* Prepare a script to deploy the application on QA environment and save it as deploy\_app\_on\_qa\_environment.sh under ansible/scripts folder.
* Commit the change, then push the script to the remote repo.

**MSP 20 - Build and Deploy App on QA Environment Manually**

* Create feature/msp-20 branch from dev.
* Create a Jenkins Job with name of build-and-deploy-petclinic-on-qa-env to build and deploy the app on QA environment manually on release branch using following script, and save the script as build-and-deploy-petclinic-on-qa-env-manually.sh under jenkins folder.
* Commit the change, then push the script to the remote repo.
* Merge dev into release branch, then run build-and-deploy-petclinic-on-qa-env job to build and deploy the app on QA environment manually.

**MSP 21 - Prepare a QA Pipeline**

* Create feature/msp-21 branch from dev.
* Create a QA Pipeline on Jenkins with name of petclinic-weekly-qa with following script and configure a cron job to trigger the pipeline every Sundays at midnight (59 23 \* \* 0) on release branch. Petclinic weekly build pipeline should be built on permanent QA environment.
* Prepare a Jenkinsfile for petclinic-weekly-qa builds and save it as jenkinsfile-petclinic-weekly-qa under jenkins folder.
* Commit the change, then push the script to the remote repo.
* Merge dev into release branch to build and deploy the app on QA environment with pipeline.