<u>Task - Build a pipeline</u>

1. Use Spring pet-clinic (https://github.com/spring-projects/spring-petclinic) as your project source code
2. Build a Jenkins pipeline with the following steps:
    a. Compile the code
    b. Run the tests
    c. Package the project as a runnable Docker image
3. Make sure all dependencies are resolved from JCenter
4. Bonus - use JFrog Artifactory in your pipeline

<u>Deliverables:</u>

1. GitHub link to the repo including
    1. Jenkins file within that repo
    2. Docker file within that repo
    3. readme.md file explaining the work and how to run the project
2. Attached runnable docker image + the command to run it

<u>Assumptions:</u>

1. Setup the Jenkins master from scratch
2. Setup the Jenkins worker Node scratch
3. Install the required dependencies and tools
    a. Setup Jfrog Artifactory for proxying the artifacts and hosting the build artifacts
    b. Install maven, JDK for code compilation, and docker for packaging

Jenkins Master Setup:

1. Used Jenkins master docker image for this setup
2. Command to launch the Jenkins Master and perform the setup
    a. ***docker run -d -p 8081:8080/tcp -p 50000:50000/tcp -v ${HOME}/Jfrog/jenkins_home:/var/jenkins_home jenkins/jenkins:lts***
3. Explanation of the above command:
    a. Start the docker container on the host where docker demon is installed and running, using the "**jenkins/jenkins:lts**" image for Jenkins master.
    b. Mapped the host and container port to access the Jenkins admin console over the browser. The host port is **8081** and the container port is **8080**
4. Verify that Jenkins master is up and running and perform the admin account setup and required plugin installation.

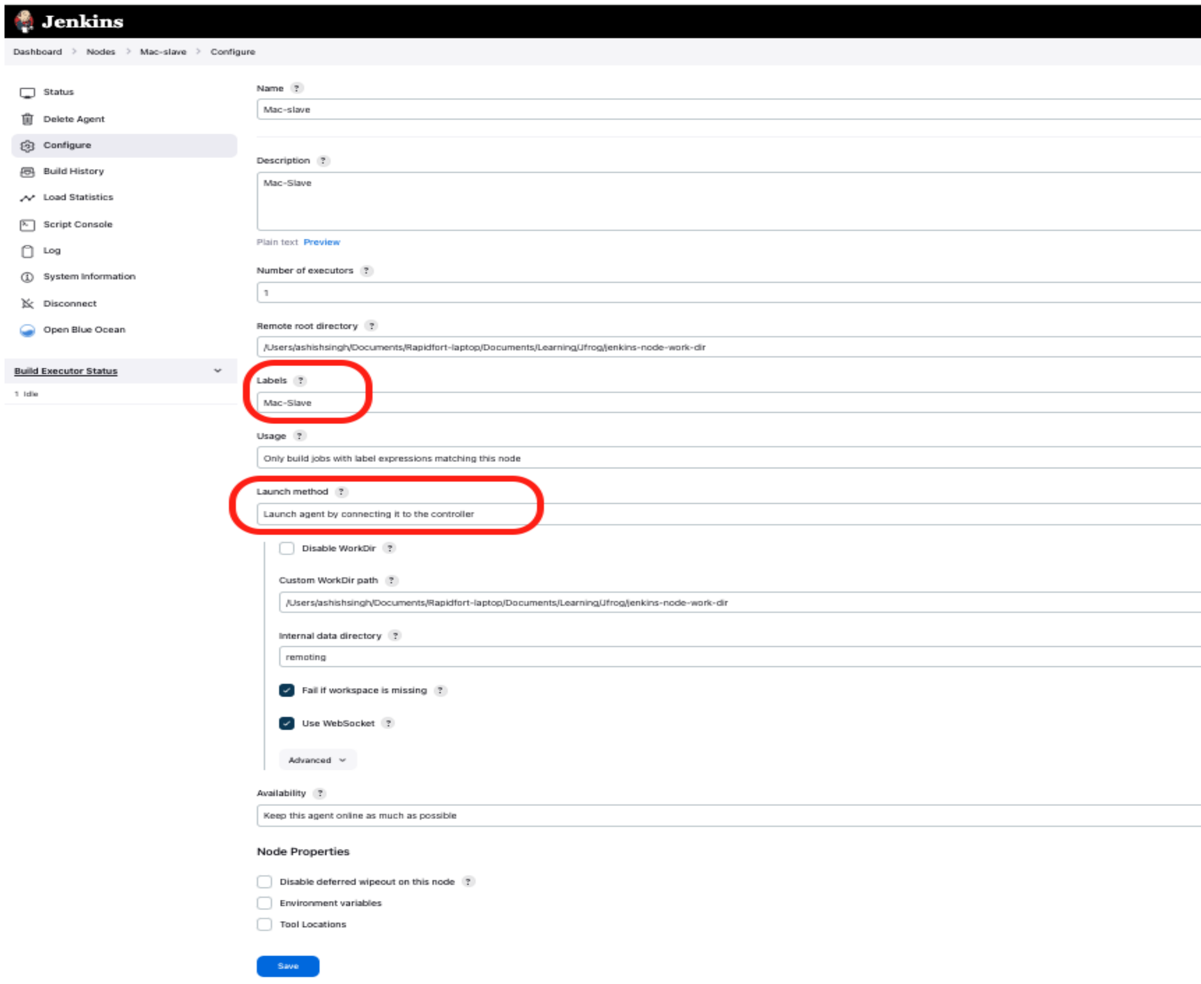
    docker ps -a | grep jenkins

```
f643427e1325   jenkins/jenkins:lts        "/usr/bin/tini -- /u…"   7 hours ago      Up 7
hours      0.0.0.0:50000->50000/tcp, 0.0.0.0:8081->8080/tcp
compassionate_mendeleev
```

5. Plugin list Refer to the Jenkins Plugin List document in the codebase.
    a. File name:

Jenkins worker node setup:

1. Since Jenkins master is running as a container, it is not advised to use the master as a worker node.
2. There are two ways to set up the Jenkins worker node:
    a. Fixed node on any cloud (Debian or Ubuntu)
    b. Container running on any container orchestration engine.
3. For Master to Worker node communication, we can setup either via:
    a. SSH or
    b. HTTP
        i. Used HTTP communication for this demo and it is better also for the Jenkins performance if we have large nodes attached to Jenkins, as the master doesn't need to maintain the configuration if the worker node is disconnected.
4. Create a Node configuration:
    a. Use the launch agent by connecting to the controller, refer to screenshots



Name: Mac-slave

Description: Mac-Slave

Number of executors: 1

Remote root directory: /Users/ashishsingh/Documents/Rapidfort-laptop/Documents/Learning/Jfrog/jenkins-node-work-dir

Labels: Mac-Slave

Usage: Only build jobs with label expressions matching this node

Launch method: Launch agent by connecting it to the controller

- [ ] Disable WorkDir

Custom WorkDir path: /Users/ashishsingh/Documents/Rapidfort-laptop/Documents/Learning/Jfrog/jenkins-node-work-dir

Internal data directory: remoting

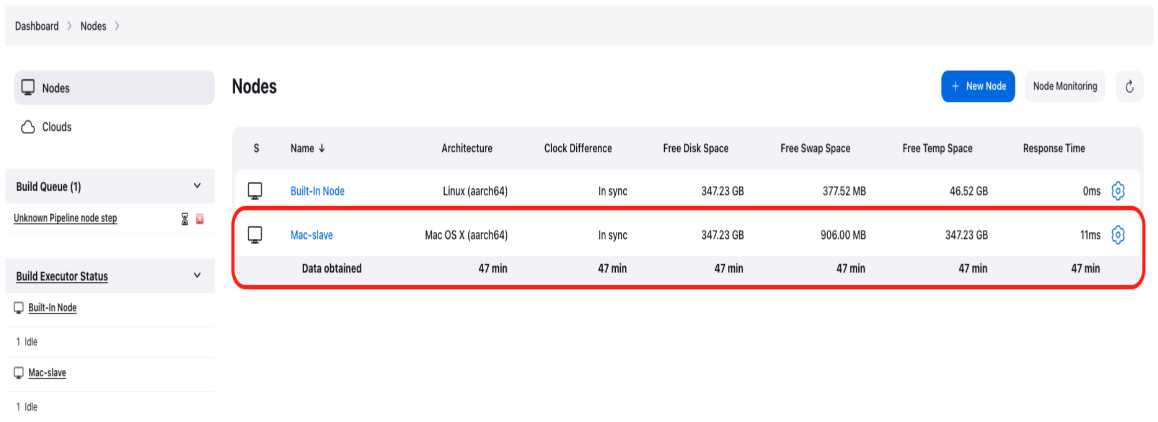
- [x] Fail if workspace is missing
- [x] Use WebSocket

Availability: Keep this agent online as much as possible

Node Properties

- [ ] Disable deferred wipeout on this node
- [ ] Environment variables
- [ ] Tool Locations

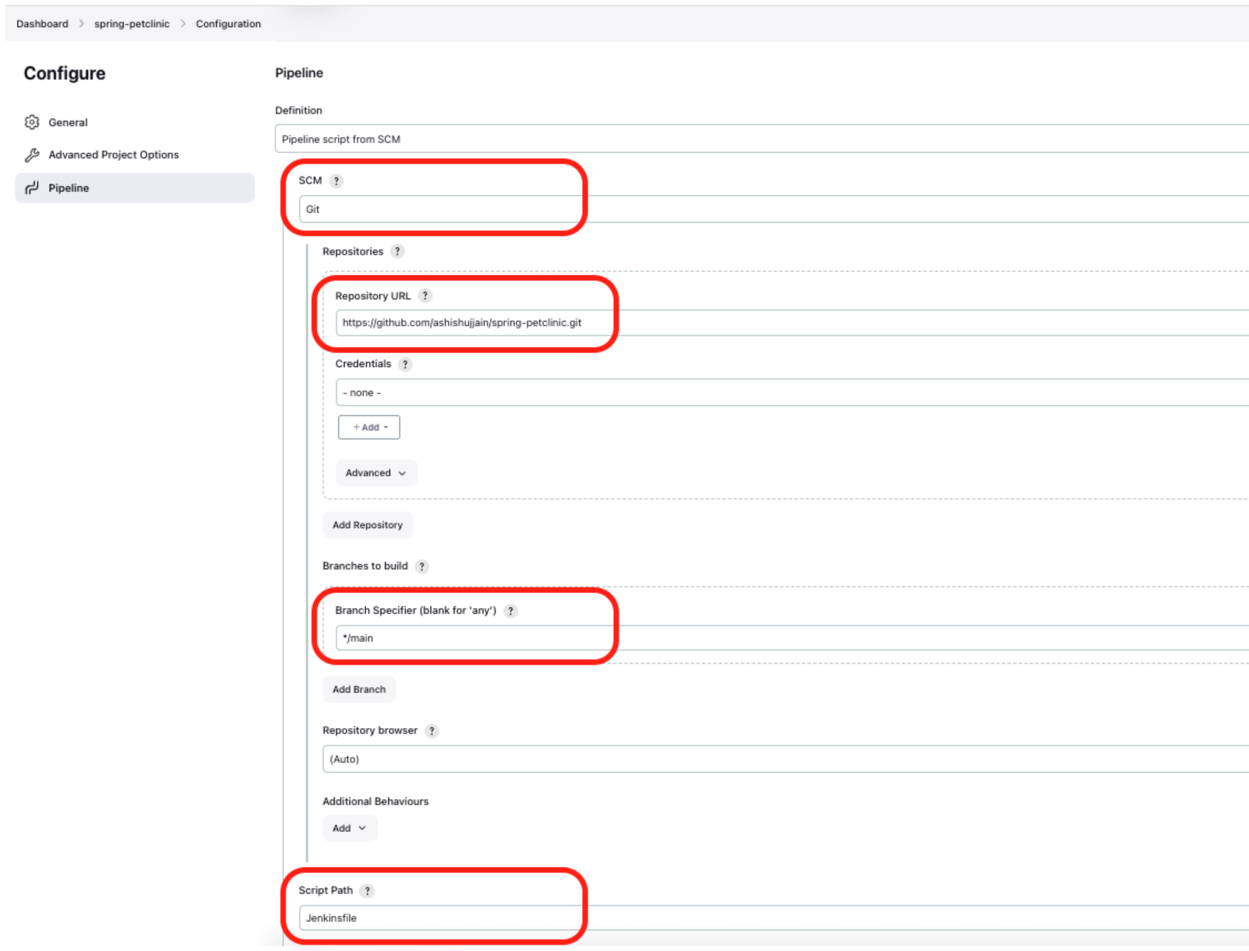
b. Download the agent jar
c. Run the command from the worker node:
   i. java -jar agent.jar -jnlpUrl http://192.168.4.36:8081/computer/Mac%2Dslave/jenkins-agent.jnlp -secret 9427f792e2a4e1d8fbf9d0d63e71d7359df61cd1ed1d6e73c67fbbc32b384dd9 -workDir "/Users/ashishsingh/Documents/Rapidfort-laptop/Documents/Learning/Jfrog/jenkins-node-work-dir" -failIfWorkDirIsMissing



S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (aarch64)	In sync	347.23 GB	377.52 MB	46.52 GB	0ms
	Mac-slave	Mac OS X (aarch64)	In sync	347.23 GB	906.00 MB	347.23 GB	11ms
	Data obtained	47 min	47 min	47 min	47 min	47 min	47 min

Jenkins Pipeline job setup:

1. Create the Jenkins pipeline job and refer to the Jenkinsfile committed in GitHub https://github.com/ashishujjain/spring-petclinic/blob/main/Jenkinsfile
   Jenkinsfile has all the stages of the pipeline, which agent label to use, and archiving the Unit Test Results.



Dashboard > spring-petclinic > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/ashishujjain/spring-petclinic.git

Credentials: - none -

Branches to build

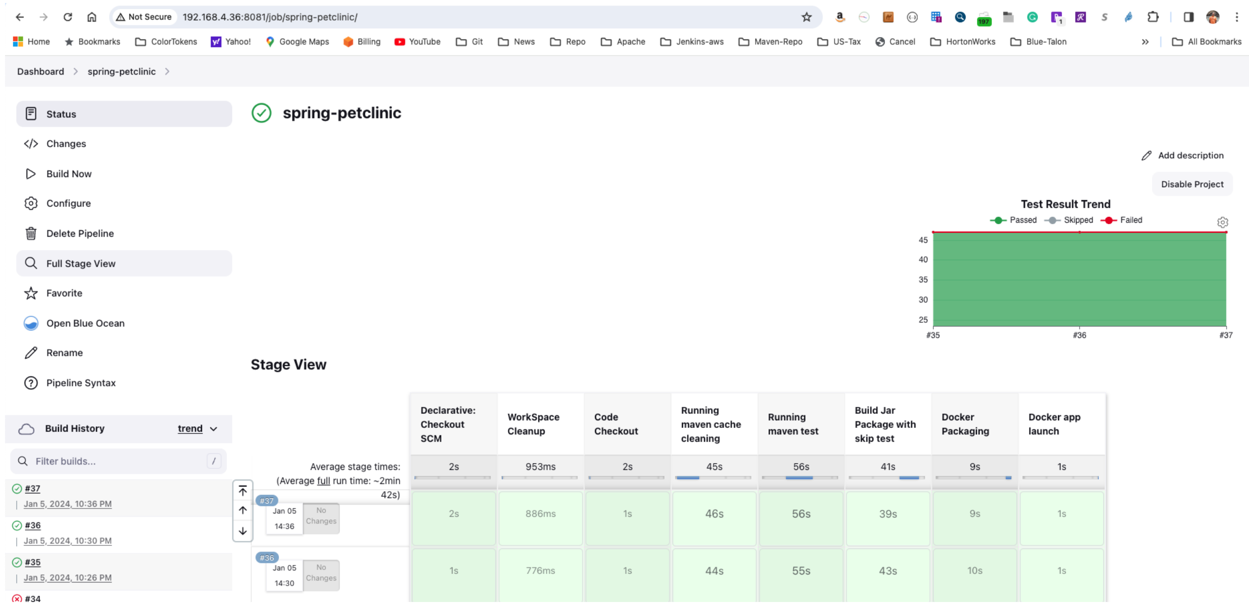
Branch Specifier (blank for 'any'): */main

Repository browser: (Auto)

Additional Behaviours

Script Path: Jenkinsfile

2. Use the open-Blue-Ocean Plugin to represent the pipeline view, refer to screenshots for Pipeline and unit test result details
3. Stages in the pipeline are as below:
    a. Cleanup workspace
    b. Code checkout
    c. Running maven cache cleaning
        i. This is done to reference the use of Jfrog setup on Jfrog Cloud
        ii. Refer to the pipeline log file for the reference of https://ashishsinghjfrog.jfrog.io
    d. Running maven test
        i. Just running the compilation and unit testing
        ii. Archiving the test results
    e. Build Jar Package with skip test
        i. Building the artifacts with skip Test
    f. Docker Packaging
        i. To share the artifacts
    g. Docker app launch
        i. Docker app launch to test the app


Not Secure 192.168.4.36:8081/job/spring-petclinic/

Dashboard > spring-petclinic >

spring-petclinic

Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Favorite
Open Blue Ocean
Rename
Pipeline Syntax

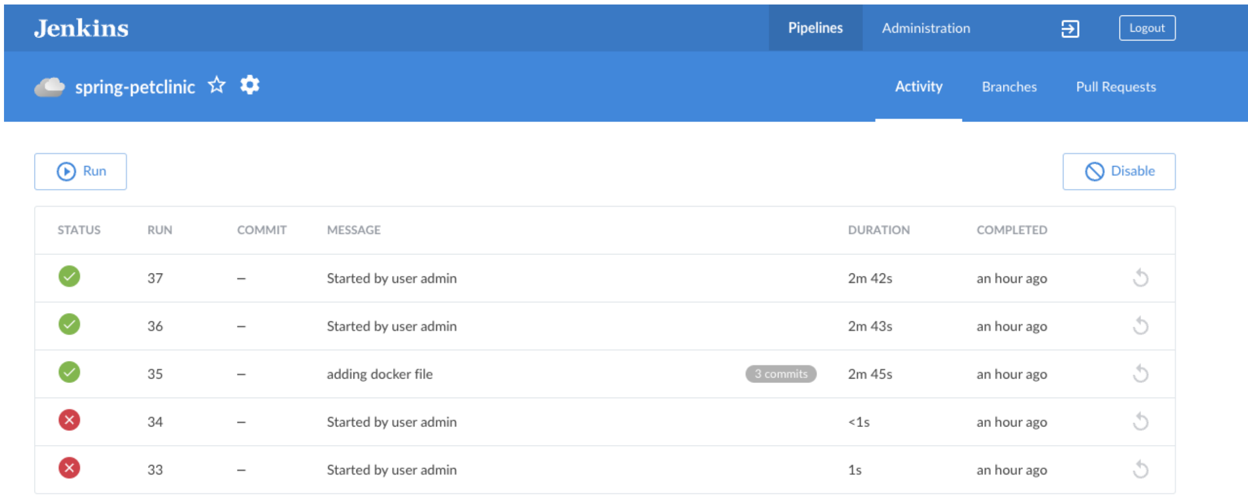
Build History trend

#37 Jan 5, 2024, 10:36 PM
#36 Jan 5, 2024, 10:30 PM
#35 Jan 5, 2024, 10:26 PM
#34

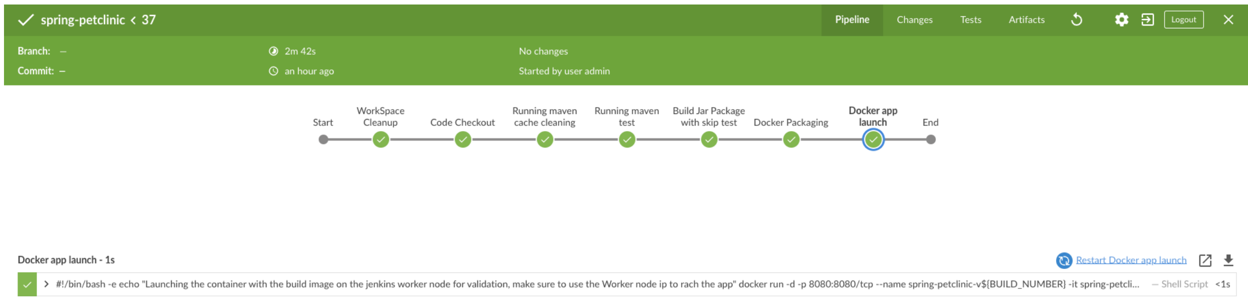
Add description
Disable Project

Stage View

	Declarative: Checkout SCM	WorkSpace Cleanup	Code Checkout	Running maven cache cleaning	Running maven test	Build Jar Package with skip test	Docker Packaging	Docker app launch
Average stage times: (Average full run time: ~2min 42s)	2s	953ms	2s	45s	56s	41s	9s	1s
#37 Jan 05 14:36 No Changes	2s	886ms	1s	46s	56s	39s	9s	1s
#36 Jan 05 14:30 No Changes	1s	776ms	1s	44s	55s	43s	10s	1s



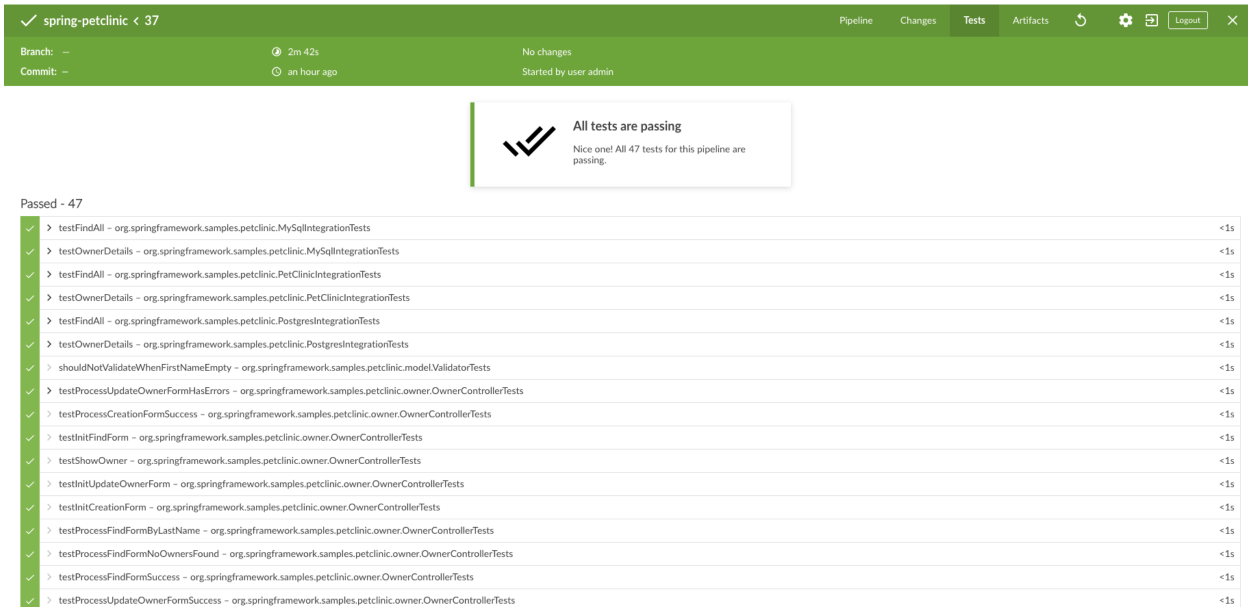
STATUS	RUN	COMMIT	MESSAGE	DURATION	COMPLETED
✓	37	–	Started by user admin	2m 42s	an hour ago
✓	36	–	Started by user admin	2m 43s	an hour ago
✓	35	–	adding docker file 3 commits	2m 45s	an hour ago
✗	34	–	Started by user admin	<1s	an hour ago
✗	33	–	Started by user admin	1s	an hour ago



Branch: –
Commit: –
2m 42s
an hour ago
No changes
Started by user admin

Docker app launch - 1s

```
#!/bin/bash -e echo "Launching the container with the build image on the jenkins worker node for validation, make sure to use the Worker node ip to rach the app" docker run -d -p 8080:8080/tcp --name spring-petclinic-v${BUILD_NUMBER} -it spring-petcli...   — Shell Script   <1s
```



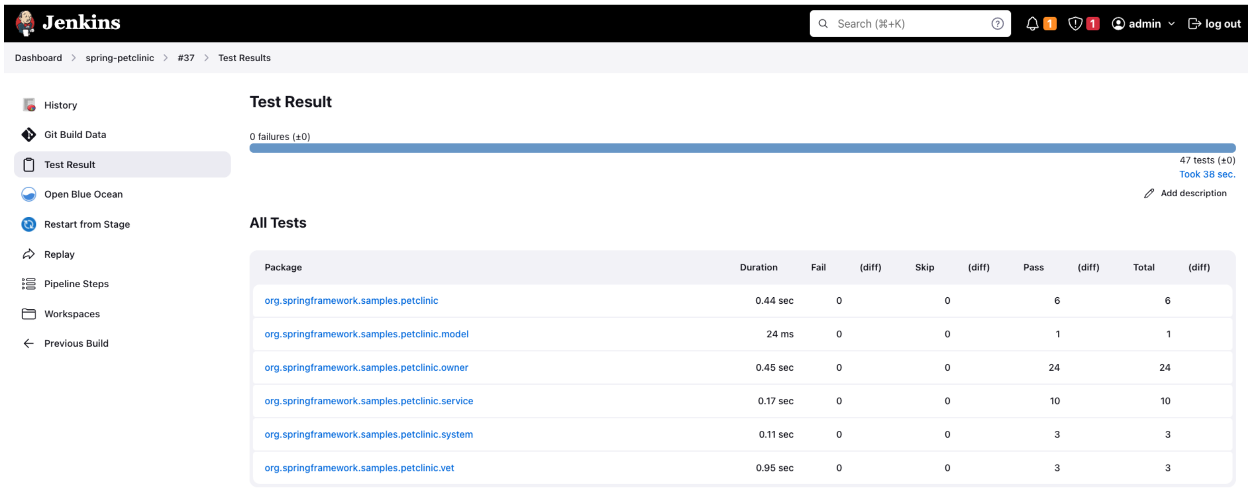
Branch: –
Commit: –
2m 42s
an hour ago
No changes
Started by user admin

All tests are passing

Nice one! All 47 tests for this pipeline are passing.

Passed - 47

```
testFindAll – org.springframework.samples.petclinic.MySqlIntegrationTests                                  <1s
testOwnerDetails – org.springframework.samples.petclinic.MySqlIntegrationTests                             <1s
testFindAll – org.springframework.samples.petclinic.PetClinicIntegrationTests                              <1s
testOwnerDetails – org.springframework.samples.petclinic.PetClinicIntegrationTests                         <1s
testFindAll – org.springframework.samples.petclinic.PostgresIntegrationTests                               <1s
testOwnerDetails – org.springframework.samples.petclinic.PostgresIntegrationTests                          <1s
shouldNotValidateWhenFirstNameEmpty – org.springframework.samples.petclinic.model.ValidatorTests           <1s
testProcessUpdateOwnerFormHasErrors – org.springframework.samples.petclinic.owner.OwnerControllerTests     <1s
testProcessCreationFormSuccess – org.springframework.samples.petclinic.owner.OwnerControllerTests          <1s
testInitFindForm – org.springframework.samples.petclinic.owner.OwnerControllerTests                        <1s
testShowOwner – org.springframework.samples.petclinic.owner.OwnerControllerTests                           <1s
testInitUpdateOwnerForm – org.springframework.samples.petclinic.owner.OwnerControllerTests                 <1s
testInitCreationForm – org.springframework.samples.petclinic.owner.OwnerControllerTests                    <1s
testProcessFindFormByLastName – org.springframework.samples.petclinic.owner.OwnerControllerTests           <1s
testProcessFindFormNoOwnersFound – org.springframework.samples.petclinic.owner.OwnerControllerTests        <1s
testProcessFindFormSuccess – org.springframework.samples.petclinic.owner.OwnerControllerTests              <1s
testProcessUpdateOwnerFormSuccess – org.springframework.samples.petclinic.owner.OwnerControllerTests       <1s
```



Test Result

0 failures (±0)

47 tests (±0)
Took 38 sec.

Add description

All Tests

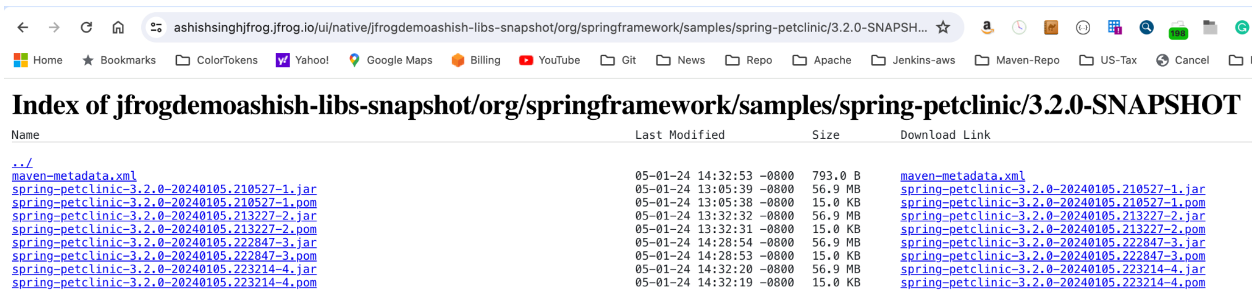
Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
org.springframework.samples.petclinic	0.44 sec	0		0		6		6	
org.springframework.samples.petclinic.model	24 ms	0		0		1		1	
org.springframework.samples.petclinic.owner	0.45 sec	0		0		24		24	
org.springframework.samples.petclinic.service	0.17 sec	0		0		10		10	
org.springframework.samples.petclinic.system	0.11 sec	0		0		3		3	
org.springframework.samples.petclinic.vet	0.95 sec	0		0		3		3	

Jfrog Setup in Cloud (Trail account):

URL : https://ashishsinghjfrog.jfrog.io/artifactory/api/maven/jfrogdemo
Setup the maven repository
Use the setting.xml from the Jfrog for the build and copy in the ~/.m2/ folder
Update the pom.xml with the below tag
https://github.com/ashishujjain/spring-petclinic/blob/b738ba49b8c04a7374f57d61efac299d572410a8/pom.xml#L40C3-L46C26

```
<distributionManagement>
    <snapshotRepository>
        <id>snapshots</id>
        <name>a0llao4piow42-artifactory-primary-0-snapshots</name>

<url>https://ashishsinghjfrog.jfrog.io/artifactory/jfrogdemoashish-libs-snapshot</url>
    </snapshotRepository>
</distributionManagement>
```

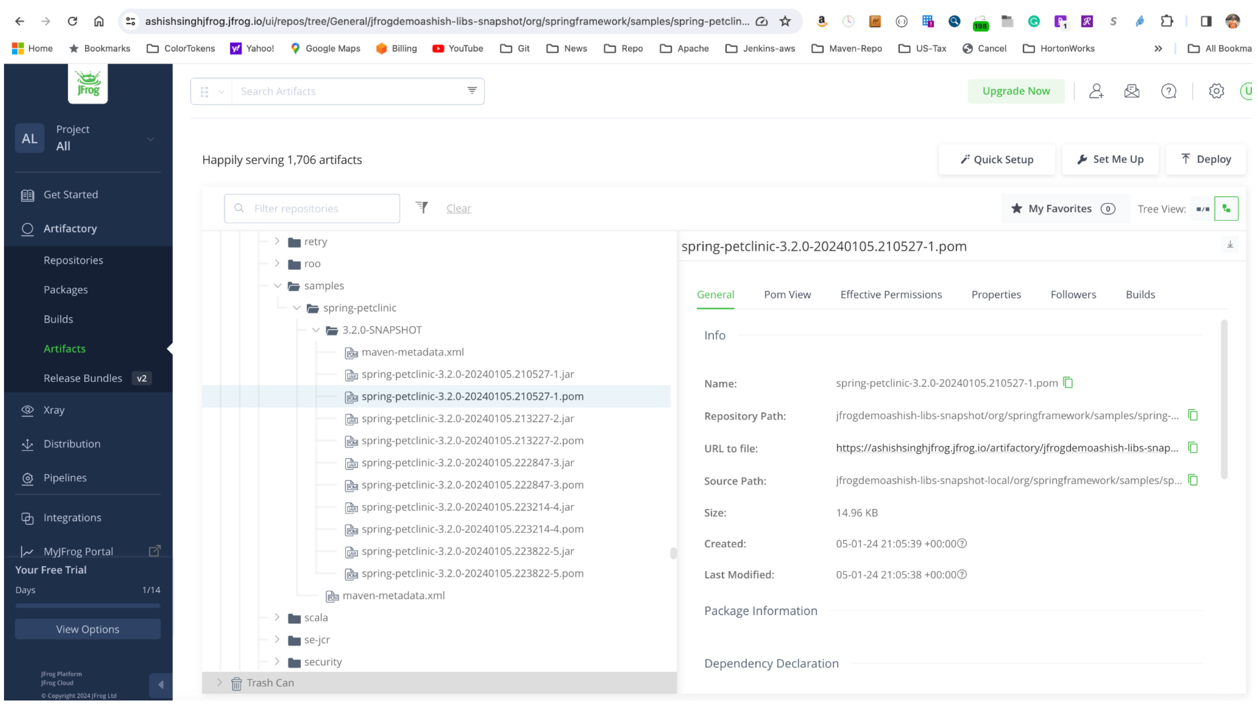


ashishsinghjfrog.jfrog.io/ui/native/jfrogdemoashish-libs-snapshot/org/springframework/samples/spring-petclinic/3.2.0-SNAPSH...
Home
Bookmarks
ColorTokens
Yahoo!
Google Maps
Billing
YouTube
Git
News
Repo
Apache
Jenkins-aws
Maven-Repo
US-Tax
Cancel

Index of jfrogdemoashish-libs-snapshot/org/springframework/samples/spring-petclinic/3.2.0-SNAPSHOT

```
Name                                                                  Last Modified         Size       Download Link

../
maven-metadata.xml                                                    05-01-24 14:32:53 -0800  793.0 B    maven-metadata.xml
spring-petclinic-3.2.0-20240105.210527-1.jar                          05-01-24 13:05:39 -0800  56.9 MB    spring-petclinic-3.2.0-20240105.210527-1.jar
spring-petclinic-3.2.0-20240105.210527-1.pom                          05-01-24 13:05:38 -0800  15.0 KB    spring-petclinic-3.2.0-20240105.210527-1.pom
spring-petclinic-3.2.0-20240105.213227-2.jar                          05-01-24 13:32:32 -0800  56.9 MB    spring-petclinic-3.2.0-20240105.213227-2.jar
spring-petclinic-3.2.0-20240105.213227-2.pom                          05-01-24 13:32:31 -0800  15.0 KB    spring-petclinic-3.2.0-20240105.213227-2.pom
spring-petclinic-3.2.0-20240105.222847-3.jar                          05-01-24 14:28:54 -0800  56.9 MB    spring-petclinic-3.2.0-20240105.222847-3.jar
spring-petclinic-3.2.0-20240105.222847-3.pom                          05-01-24 14:28:53 -0800  15.0 KB    spring-petclinic-3.2.0-20240105.222847-3.pom
spring-petclinic-3.2.0-20240105.223214-4.jar                          05-01-24 14:32:20 -0800  56.9 MB    spring-petclinic-3.2.0-20240105.223214-4.jar
spring-petclinic-3.2.0-20240105.223214-4.pom                          05-01-24 14:32:19 -0800  15.0 KB    spring-petclinic-3.2.0-20240105.223214-4.pom
```



ashishsinghjfrog.jfrog.io/ui/repos/tree/General/jfrogdemoashish-libs-snapshot/org/springframework/samples/spring-petclin...

Home Bookmarks ColorTokens Yahoo! Google Maps Billing YouTube Git News Repo Apache Jenkins-aws Maven-Repo US-Tax Cancel HortonWorks All Bookma

Happily serving 1,706 artifacts

```
retry
roo
samples
  spring-petclinic
    3.2.0-SNAPSHOT
      maven-metadata.xml
      spring-petclinic-3.2.0-20240105.210527-1.jar
      spring-petclinic-3.2.0-20240105.210527-1.pom
      spring-petclinic-3.2.0-20240105.213227-2.jar
      spring-petclinic-3.2.0-20240105.213227-2.pom
      spring-petclinic-3.2.0-20240105.222847-3.jar
      spring-petclinic-3.2.0-20240105.222847-3.pom
      spring-petclinic-3.2.0-20240105.223214-4.jar
      spring-petclinic-3.2.0-20240105.223214-4.pom
      spring-petclinic-3.2.0-20240105.223822-5.jar
      spring-petclinic-3.2.0-20240105.223822-5.pom
    maven-metadata.xml
scala
se-jcr
security
```

spring-petclinic-3.2.0-20240105.210527-1.pom

General | Pom View | Effective Permissions | Properties | Followers | Builds

Info

Name:	spring-petclinic-3.2.0-20240105.210527-1.pom
Repository Path:	jfrogdemoashish-libs-snapshot/org/springframework/samples/spring-...
URL to file:	https://ashishsinghjfrog.jfrog.io/artifactory/jfrogdemoashish-libs-snap...
Source Path:	jfrogdemoashish-libs-snapshot-local/org/springframework/samples/sp...
Size:	14.96 KB
Created:	05-01-24 21:05:39 +00:00
Last Modified:	05-01-24 21:05:38 +00:00

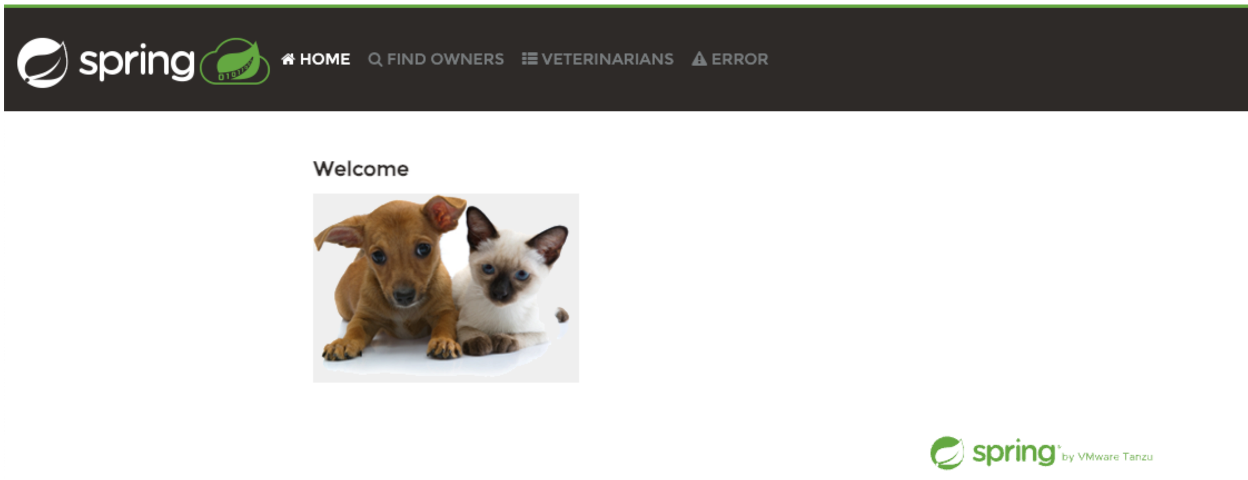
Package Information

Dependency Declaration

How to Run the docker image of spring-projects/spring-petclinic app.

1. Download the docker image spring-petclinic_v37.tar, shared via Google Drive (**not able to share via GitHub**) https://drive.google.com/file/d/1534zq3zvf30TH4IIrJGnM_OuWFsq-VrC/view?usp=sharing
2. Load the image file to local docker registry
   a. docker load -i spring-petclinic_v37.tar
3. Run the app on the host running docker
   a. docker run -d -p 8080:8080/tcp --name spring-petclinic-v37 -it spring-petclinic-v37:v37

4. Access the App using http://<host ip>:8080 (host port 8080)



Final deliverables:

GitHub link to the repo including

1. Jenkins file within that repo :
https://github.com/ashishujjain/spring-petclinic/blob/main/Jenkinsfile
2. Docker file within that repo:
https://github.com/ashishujjain/spring-petclinic/blob/main/Dockerfile
3. readme.md file explaining the work and how to run the project

2. Attached runnable docker image + the command to run it
https://drive.google.com/file/d/1534zq3zvf30TH4IIrJGnM_OuWFsq-VrC/view?usp=sharing