**DevOps Position Hands-on Test Solution**

# High-Level Design

1. Install Oracle VirtualBox
2. Download CentOs 8
3. Create VM and Install CentOs 8
4. Upgrade System
5. Expose ports: 22 and 8080
6. Enable SSH Access on the VM
7. Install Package: python3
8. Install Packages: docker, docker-compose
9. Verify Docker by Running hello-world Image
10. Create User for Jenkins
11. Copy Files to Host
12. Jenkins Project Details
13. Build Jenkins Image
14. Run Jenkins Image as Container
15. Jenkins Jobs

# Detailed Solution

## Install Oracle VirtualBox

1. Download installer from: <https://download.virtualbox.org/virtualbox/6.1.30/VirtualBox-6.1.30-148432-Win.exe>
2. Install VirtualBox

## Download CentOs 8

1. Download rpm from: <http://centos.spd.co.il/8.5.2111/isos/x86_64/CentOS-8.5.2111-x86_64-dvd1.iso>

## Create VM and Install CentOs 8

1. Create VM named: CentOs8, Memory Size: 4096 MB, File Size: 100 GB, Operating System Type: Linux, Operating System Version: Red Hat (64 bit)
2. Update Settings to use Storage 🡪 Optical Drive: the downloaded .iso file
3. Start the VM
4. Install OS including setup of root user

## Upgrade System

1. Open Terminal and execute:

[root@localhost ~]# yum -y update && yum install -y \

yum-utils \

gcc \

openssl-devel \

bzip2-devel \

libffi-devel \

zlib-devel \

readline-devel \

tk-devel \

tcl-devel \

sqlite-devel \

wget \

unzip \

zip \

less \

sudo \

net-tools \

bind-utils \

tcsh \

passwd \

which \

openssh \

openssh-clients \

openssh-server \

openssl \

openssl-libs \

java-1.8.0-openjdk \

epel-release \

jq \

nfs-utils \

tar

## Expose Ports: 22 and 8080

1. Since I’m using Network Adapter type: NAT, I need to port forwarding.
2. Open Terminal and execute:

[root@localhost ~]# ip addr show

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127**.**0**.**0**.**1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc fq\_codel state UP group default qlen 1000

link/ether 08:00:27:da:53:e9 brd ff:ff:ff:ff:ff:ff

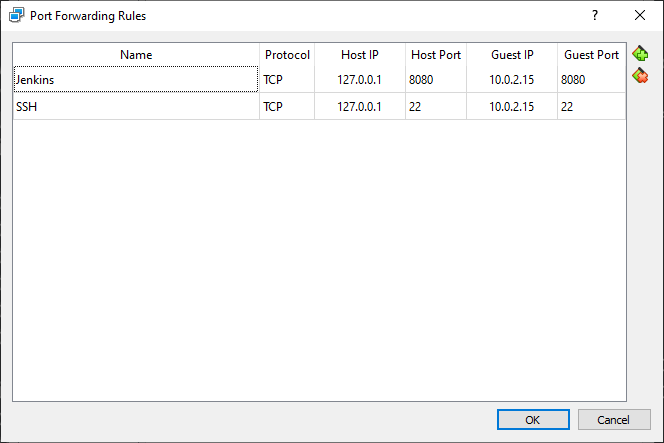
inet 10**.**0**.**2**.**15/24 brd 10**.**0**.**2**.**255 scope global dynamic noprefixroute enp0s3

valid\_lft 84393sec preferred\_lft 84393sec

inet6 fe80::a00:27ff:feda:53e9/64 scope link noprefixroute

valid\_lft forever preferred\_lft forever

1. Getting the IPv4 of the enp0s3 network: 10.0.2.15
2. Open VirtualBox Settings 🡪 Network 🡪 Advanced 🡪 Port Forwarding
3. Setup as below:



1. Reboot system

## Enable SSH Access on the VM

1. Open Terminal and execute:

[root@localhost ~]# systemctl status sshd

1. If inactive, execute:

[root@localhost ~]# systemctl start sshd

1. Check port by execute:

[root@localhost ~]# cat /etc/ssh/sshd\_config | grep Port

1. If the port is different from 22, change it and restart service
2. Finally, enable sshd to start automatically after each system reboot by execute:

[root@localhost ~]# systemctl enable sshd

1. Back to laptop, open Command Line and execute:

C:\Users\mronen\Downloads>ssh -p 22 root@localhost**.**localdomain

root@localhost**.**localdomain's password:

Activate the web console with: systemctl enable --now cockpit.socket

Last login: Fri Jan 14 14:09:32 2022

[root@localhost ~]#

1. SSH Connection Established

## Install Package: Python3

1. Open Terminal and execute:

[root@localhost ~]# cd /usr/src

[root@localhost ~]# sudo wget https://www.python.org/ftp/python/3.7.12/Python-3.7.12.tgz

[root@localhost ~]# tar xzf Python-3.7.12.tgz

[root@localhost ~]# cd /usr/src/Python-3.7.12

[root@localhost ~]# ./configure --enable-optimizations

**.**

**.**

**.**

[root@localhost ~]# make altinstall

**.**

**.**

**.**

[root@localhost ~]# rm -rfv /usr/src/Python-3.7.12.tgz

[root@localhost ~]# ln -s /usr/local/bin/python3.7 /usr/bin/python3.7

[root@localhost ~]# ln -s /usr/local/bin/python3.7 /usr/bin/python3

[root@localhost ~]# /usr/bin/python3 --version

Python 3**.**7**.**12

[root@localhost ~]# /usr/bin/python3 -m pip install --upgrade pip==20.2.4

## Install Packages: docker, docker-compose

1. Install docker

[root@localhost ~]# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

[root@localhost ~]# yum install docker-ce docker-ce-cli containerd.io

[root@localhost ~]# systemctl start docker

1. Verify docker

[root@localhost ~]# docker --version

Docker version 20**.**10**.**12, build e91ed57

1. Execute docker Linux post installation (Reference: [link](https://docs.docker.com/engine/install/linux-postinstall/))
2. Install docker-compose (Reference: [link](https://docs.docker.com/compose/install/))

[root@localhost ~]# curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

[root@localhost ~]# chmod +x /usr/local/bin/docker-compose

[root@localhost ~]# ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

1. Verify docker-compose

[root@localhost ~]# docker-compose --version

docker-compose version 1**.**29**.**2, build 5becea4c

## Verify Docker by Running hello-world Image

1. Verify docker by running hello-world image (from non-root user: mronen)

[mronen@localhost ~]# docker run hello-world

Unable to find image 'hello-world:latest' locally

latest: Pulling from library/hello-world

2db29710123e: Pull complete

Digest: sha256:975f4b14f326b05db86e16de00144f9c12257553bba9484fed41f9b6f2257800

Status: Downloaded newer image **for** hello-world:latest

Hello from Docker**!**

This message shows that your installation appears to be working correctly**.**

To generate this message, Docker took the following steps:

1**.** The Docker client contacted the Docker daemon**.**

2**.** The Docker daemon pulled the "hello-world" image from the Docker Hub**.**

(amd64)

3**.** The Docker daemon created a new container from that image which runs the

executable that produces the output you are currently reading**.**

4**.** The Docker daemon streamed that output to the Docker client, which sent it

to your terminal**.**

To try something more ambitious, you can run an Ubuntu container with:

$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:

https://hub.docker.com/

For more examples and ideas, visit:

https://docs.docker.com/get-started/

## Create User for Jenkins

1. Open Terminal and execute:

[jenkins@localhost ~]$ useradd Jenkins

[jenkins@localhost ~]$ passwd jenkins

1. Add user Jenkins to sudoers file /etc/sudoers by executing:

[jenkins@localhost ~]$ sudo visudo

Edit file and add entry:

jenkins ALL=(ALL) ALL

1. Create folder under ~/jenkins named: jenkins\_home to store Jenkins data.

Container Jenkins user uid is 1000. Set the new created folder ~/jenkins/jenkins\_home to uid 1000 (Reference: [link](https://hub.docker.com/_/jenkins))

[jenkins@localhost ~]$ cd /home/jenkins

[jenkins@localhost ~]$ mkdir jenkins\_home

[jenkins@localhost ~]$ cd jenkins\_home

[jenkins@localhost ~]$ sudo chown -R 1000:1000 .

1. Create folder under: /data named: test to store the output of job #9 (mount point)

[root @localhost ~]$ cd /data

[root@localhost ~]$ mkdir test

## Copy Files to host

1. Copy the folder: my-jenkins into the centos machine at: /home/jenkins/my-jenkins

C:\>scp C:\GitHubRepos\docker-exercise\my-jenkins\\* jenkins@localhost.localdomain:/home/jenkins/my-jenkins

jenkins@localhost.localdomain's password:

Dockerfile 100% 2788 557.9KB/s 00:00

docker-compose.yml 100% 577 115.5KB/s 00:00

entrypoint.sh 100% 126 25.2KB/s 00:00

jenkins.yaml 100% 4373 624.7KB/s 00:00

plugins.txt 100% 2333 291.8KB/s 00:00

## Jenkins Project Details

I create project that contains:

1. Dockerfile to build Jenkins from: <https://hub.docker.com/r/jenkins/jenkins>
2. docker-compose.yml that defines exposed ports, container name, volumes and environment variables
3. jenkins.yaml that defines entire Jenkins configuration including credentials to my GutHub (temporary token that will expire within 7 days) and jobs definition as DSL.
4. plugins.txt that defines all plugins to install

## Build Jenkins Image

1. Open Terminal (as jenkins) and execute:

[jenkins@localhost ~]$ docker-compose build --pull --compress > my-jenkins-build**.**log

1. my-jenkins-build.log attached to this folder with name: my-jenkins-build.log
2. List of images:

[jenkins@localhost my-jenkins]$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

my-jenkins\_jenkins latest fb6f2148b934 7 minutes ago 1**.**98GB

jenkins/jenkins lts 9aee0d53624f 5 days ago 441MB

python 3 a5d7930b60cc 4 weeks ago 917MB

## Run Jenkins Image as Container

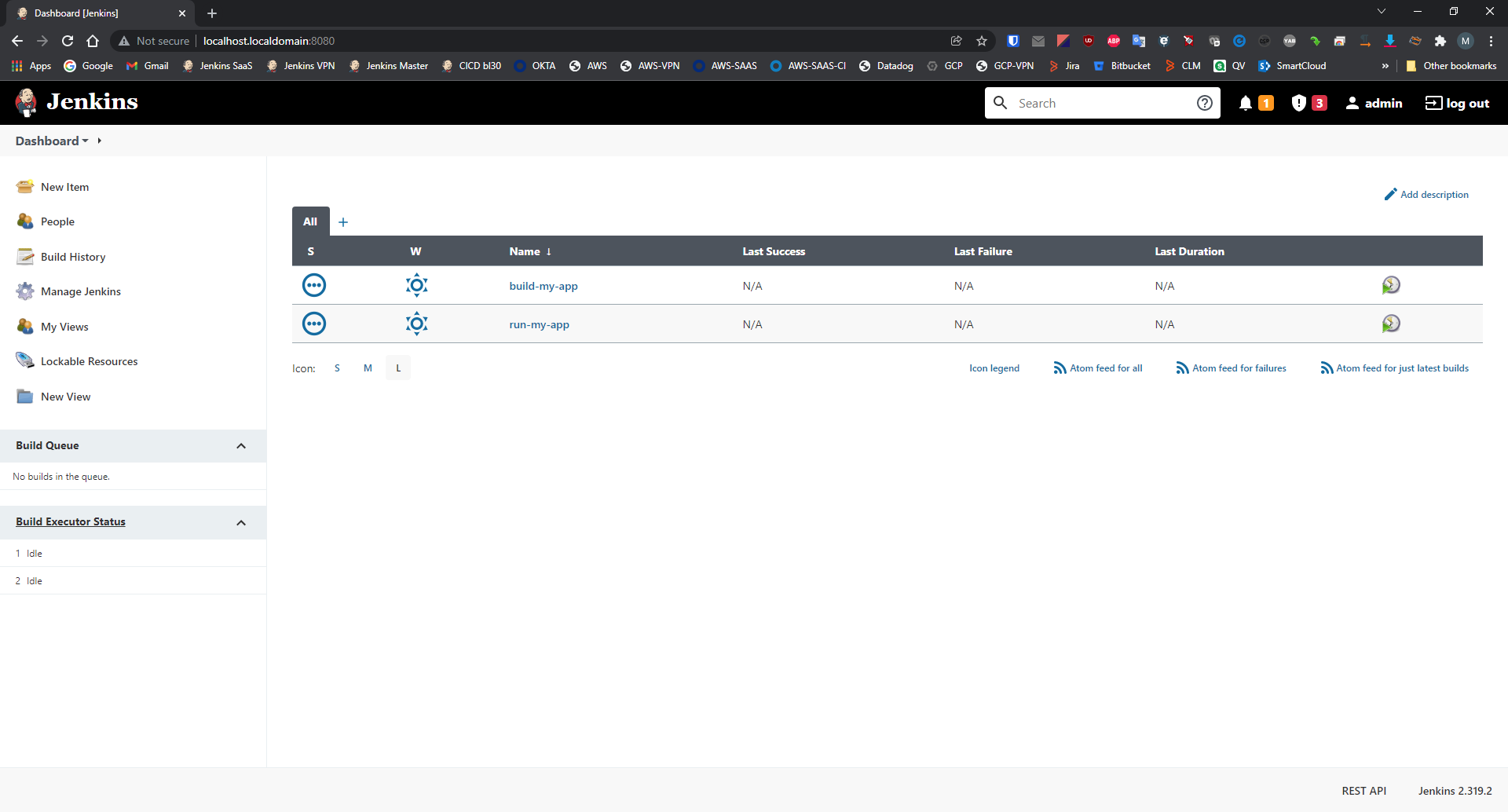
1. Open Terminal (as jenkins) and execute:

[jenkins@localhost my-jenkins]$ docker-compose up -d

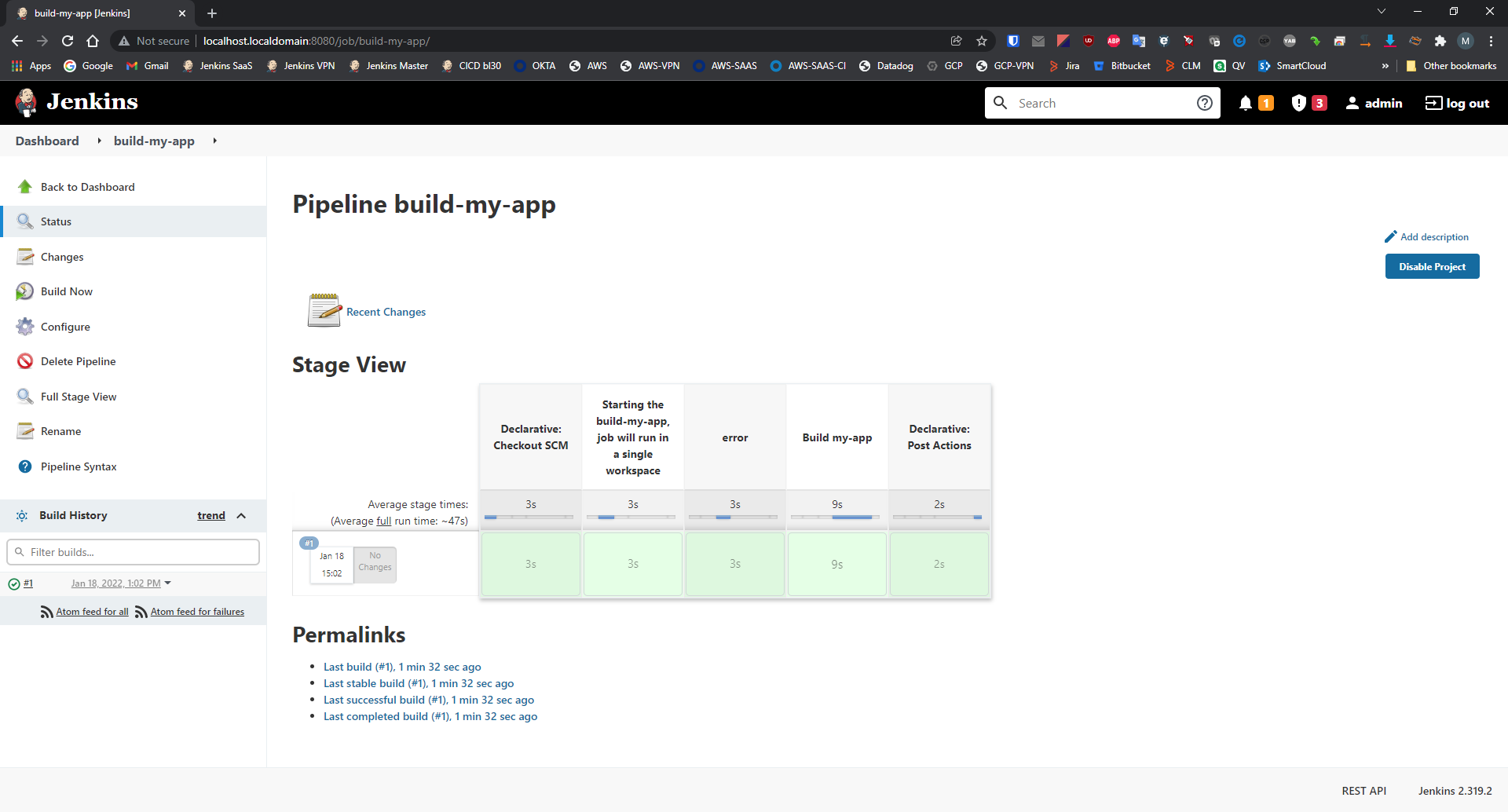
Creating my-jenkins **...** **done**

## Jenkins Jobs

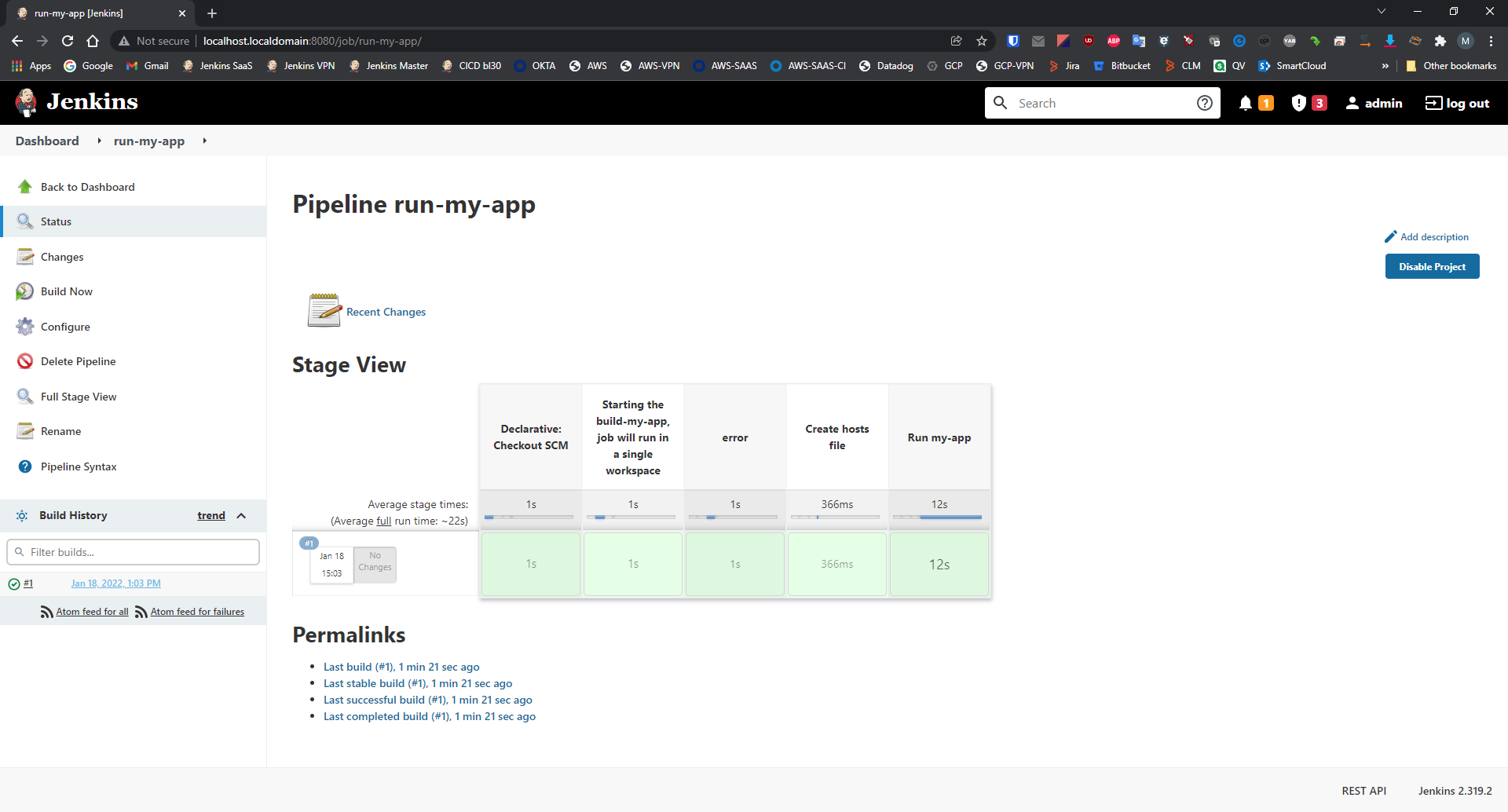
1. Login: admin/password
2. My new Jenkins already has 2 jobs:
   1. build-my-app: job to build my-app image using docker-compose with post actions on success with trigger job run-my-app
   2. run-my-app: job to run my-app image as container using Ansible
   3. Both jobs are configured as “Pipeline script from SCM” using my personal GitHub to pull Jenkinsfiles
   4. Jenkinsfiles resides on: jenkins-jobs



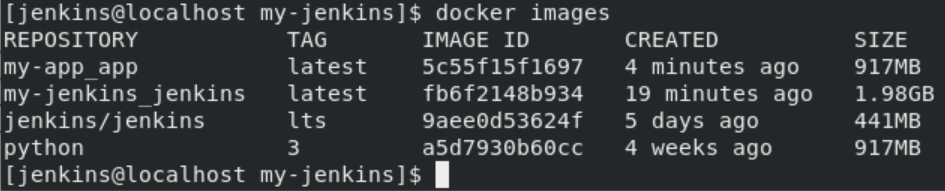
1. Manual trigger first job: build-my-app
2. Console output attached to this folder with name: build-my-app.log



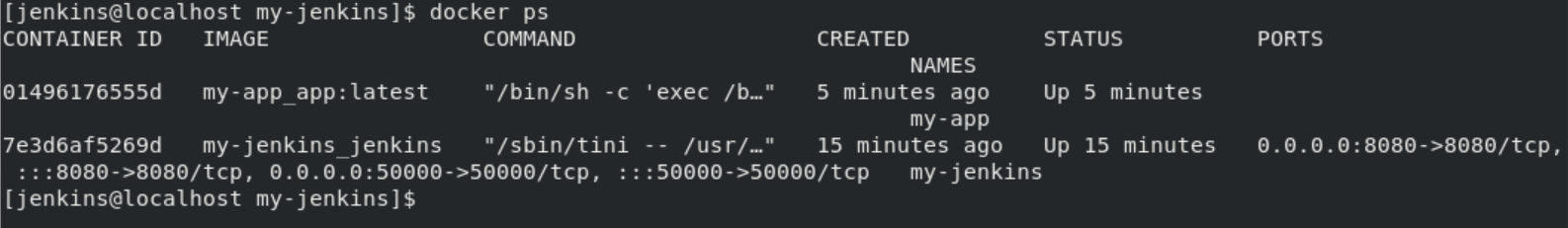
1. Automatic trigger second job: run-my-app
2. Console output attached to this folder with name: run-my-app.log



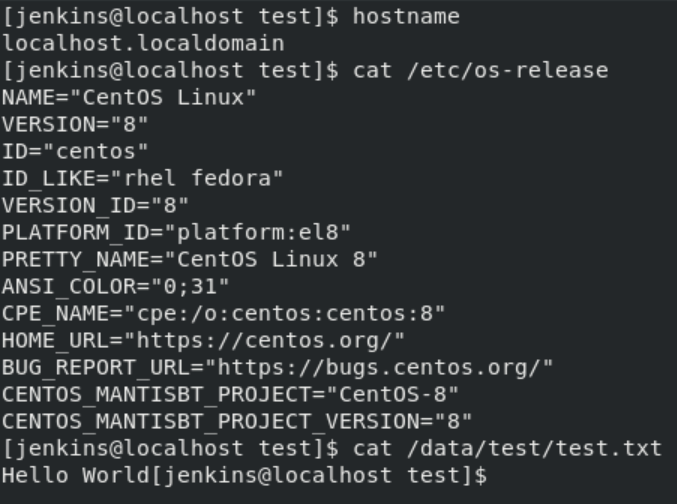
1. Docker Images after Jobs Run



1. Docker Containers after Jobs Run



1. Python script that prints “Hello World” to test.txt resides on: my-app/my-app.py
2. Dockerfiles resides on:
   1. my-jenkins/Dockerfile
   2. my-app/Dockerfile
3. Ansible script resides on: my-app/my-app-playbook.yaml
4. Content of /data/test/test.txt on my CentOs VM



## Conclusion

I do my best to prepare this solution, I learned a lot while working on this solution (Ansible (for dummies in my case), Jenkins configuration via “Configuration as a Code” plugin and docker compose.

I hope this solution will satisfy you while taking the decision on this position.

I’ll be glad to have another meeting so I can explain my decisions and implementation.

Thanks a lot, and Enjoy,

Moshe Ronen