PODMAN CI/CD WITH BASH IN DMZ
Karin I.E.

Karin Iliia Eduardovich - Master's degree in Information Systems in Economics and Management, DevOps Engineer with mor,. INFRASTRUCTURE DEPARTMENT, INVENT INC., DUBAI, UAE.

Abstract: *the article describes the use of Podman technology from RedHat corporation and shows an example of this automated deployment into a DMZ Virtual Machine.*
Keywords: *podman, bash, virtual machine, information and technology, docker, devops, software development.*

PODMAN CI/CD С BASH В DMZ
Карин И.Е.

Карин Илья Эдуардович - Магистр информационных систем в экономике и менеджменте, DevOps-инженер, отдел инфраструктуры, INVENT INC.,
г. Дубай, Объединенные Арабские Эмираты

Аннотация: *в статье описано использование технологии Podman от корпорации RedHat и показан пример такого автоматизированного развертывания в виртуальной машине DMZ.*
Ключевые слова: *podman, bash, виртуальная машина, информация и технологии, докер, devops, разработка программного обеспечения.*

Docker is an open-source platform that automates the process of deploying, managing, and running applications in containers. Containers are lightweight, self-contained, and portable environments that contain an application and all its dependencies, including libraries and other components needed to run it.

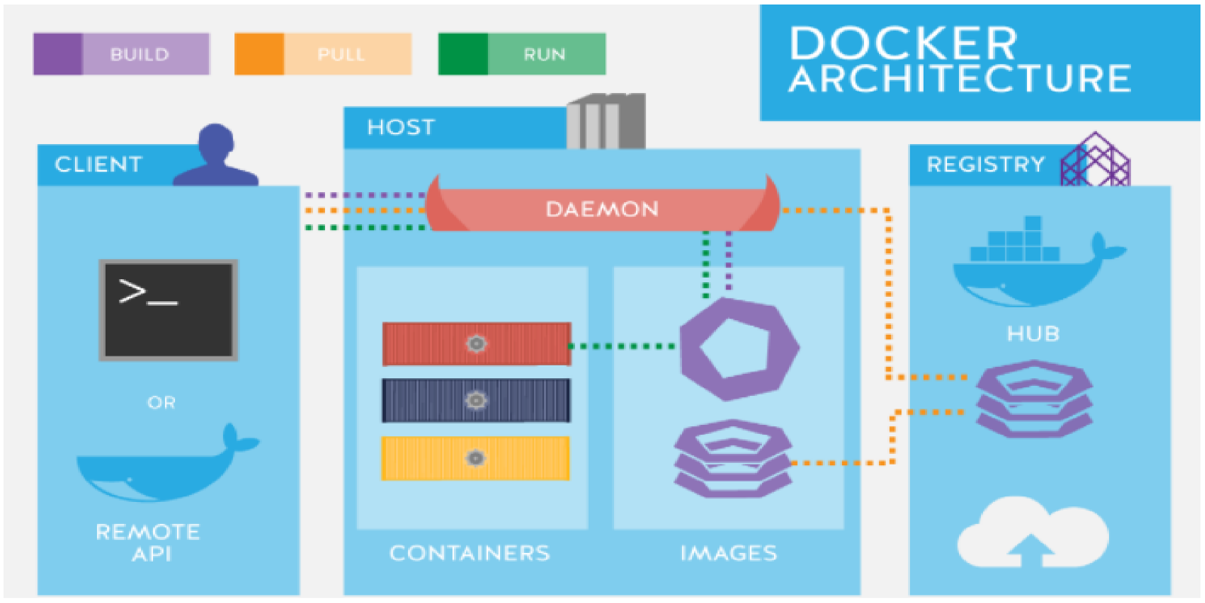
The main components of Docker technology:

Docker Engine: The Docker core that manages containers. It includes a daemon that runs on the host machine and a set of APIs for interacting with Docker.

Docker Images: Images are the templates from which containers are created. An image contains everything needed to run the application: binaries, libraries, environment variables, configuration files, etc.

Docker Containers: Containers are instances of running Docker images. They are isolated processes that run inside a shared environment (container). Each container has its own file system, network, and processes, but they use the host machine kernel.

Docker Registry: The Docker Registry is a centralized repository of Docker images. It allows users to share their created images and access images created by others.



Benefits of Docker:

Portability: Containers enable high portability of applications. The application and its dependencies are packaged in a container that can run on any platform where Docker is installed.

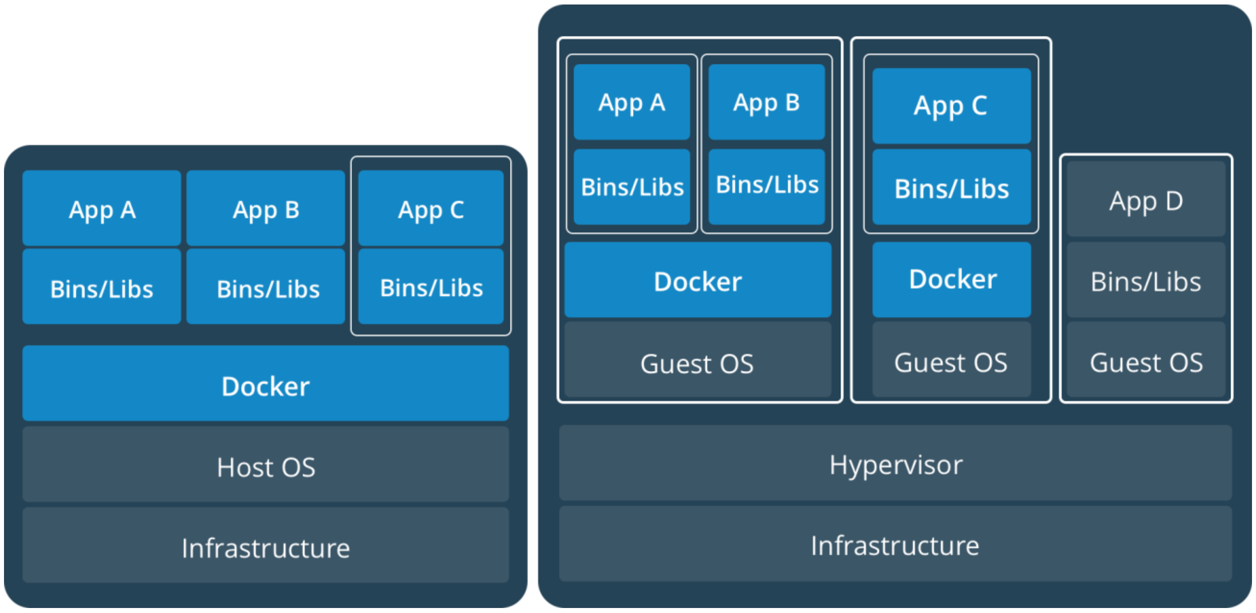
Isolation: Each container runs in an isolated environment, which prevents conflicts between applications and ensures security.

Resource efficiency: Containers share a common host machine kernel, which saves resources and significantly reduces container startup and shutdown times.

Scalability: Docker makes it easy to scale applications by allowing you to run multiple containers of the same application on a single machine or distribute them across multiple servers.

Version management: Using containers simplifies application version management by allowing you to quickly switch between different versions.

Docker is one of the most popular technologies for application development, testing, and deployment because it greatly simplifies and accelerates the software development and maintenance process.



What's wrong with Docker?

Resource sharing: Containers share a common host operating system kernel and share resources with other containers. This means that if one container is compromised, it may be easier for an attacker to move to other containers running on the same host.

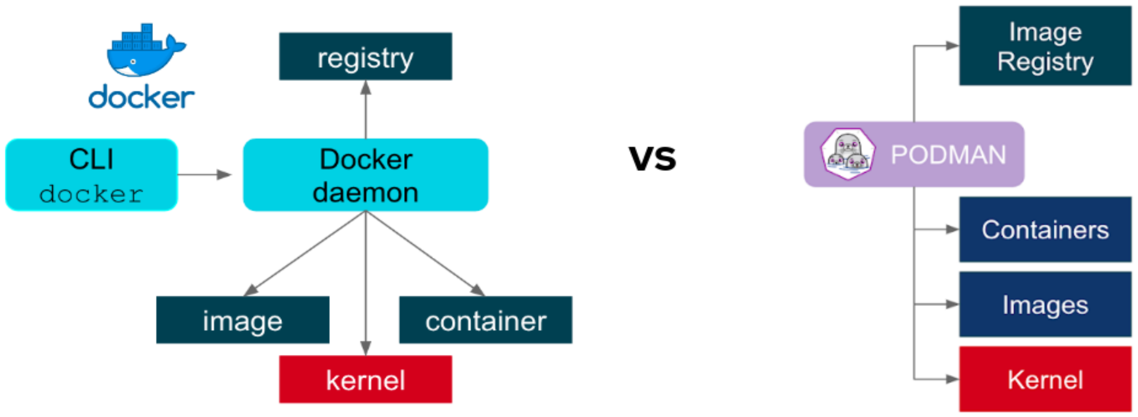
Container privileges: Incorrectly configured container privileges can allow an attacker to gain access to a host's system resources. It is important to monitor what privileges are granted to the container and minimize them.

Network Isolation: By default, Docker containers have access to the host network, which can be dangerous, if the host network does not really need to be accessible from the container. It is necessary to carefully configure the network isolation of containers.

Linux kernel features: Docker utilizes Linux kernel features, and incorrect configuration or exploitation of kernel vulnerabilities can affect container security.

Summarizing this section, we can say that the main problem with Docker is security, and it is a very serious problem that can lead to the compromise of the whole company's system.

Podman is not the only one, but the best.



Daemonless: One of the main differences between Podman and Docker is that Podman does not require a constantly running daemon (service). This significantly reduces the potential attack vector.

Support for rootless containers: Podman allows you to run containers in rootless mode, which increases security because containers do not have root privileges on the host system, and an exit attack from the container to the host becomes impossible.

Pods management: Podman supports the concept of "pods", which allows containers to be grouped into isolated groups, making them easier to manage and communicate with each other.

Flexible container management: Podman allows containers to be run as a non-administrator user.

OCI support: Podman adheres to the Open Container Initiative (OCI) standards to ensure compatibility with other OC tools, such as Docker, Kubernetes, and others.

But it's not all smooth sailing.

Lack of community and ecosystem: Compared to Docker, Podman has a smaller community of users and developers and a smaller selection of third-party tools and support resources. Finding answers to your questions, and instructions is much more difficult than with Docker.

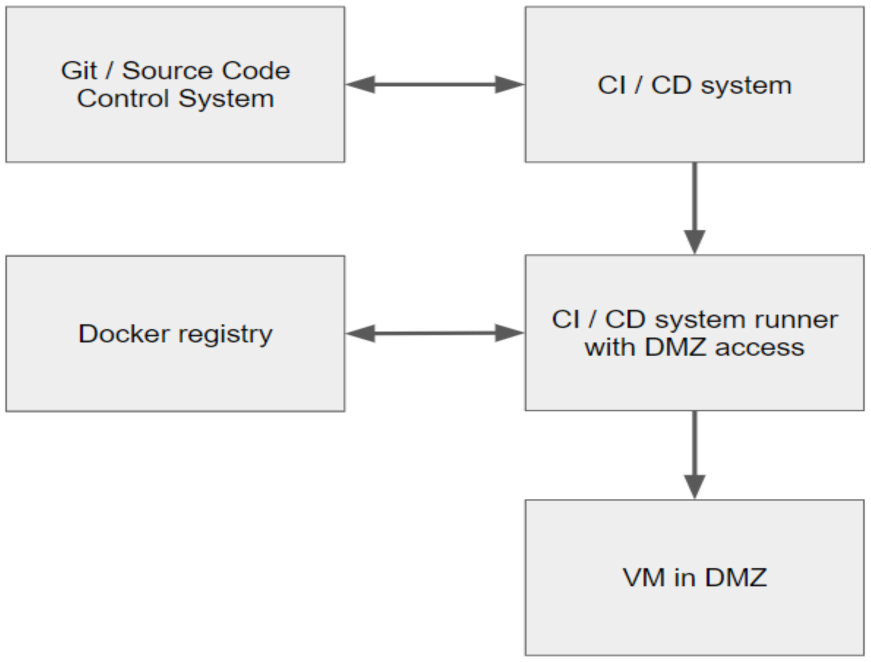
Limited Windows and Mac support: Podman is designed primarily for Linux environments, and there is very limited support for Windows and Mac OS.

So what can be built with Podman?

As an example, I will cite one of the past projects, it may seem unattractive or outdated, as Podman is developing quite actively, but current projects can not be told to the readers for the same reason we can’t use docker.

And so, we will consider a very important transactional service that should be as secure as possible.

Since Podman is not the most convenient way to deploy a distributed application, we will use Podman-compose.


Git / Source Code Control System
CI / CD system
Docker registry
CI / CD system runner with DMZ access
VM in DMZ

Prerequisites:

The most difficult thing is to configure the final VM, since the machine is in DMZ, we can only run podman-compose from a user with minimal permissions, and we will have to make the deployment and file transfers from a user with no rights to run podman, and of course both users can not have root privileges, we can configure commands with elevated rights to run in sudoers by specifying specific services to interact without entering sudo password.

Example:

```
%podman-admin ALL=(podman) NOPASSWD: /bin/systemctl restart APPlication
%podman-admin ALL=(podman) NOPASSWD: /bin/systemctl stop APPlication
%podman-admin ALL=(podman) NOPASSWD: /bin/systemctl start APPlication
%podman-admin ALL=(podman) NOPASSWD: /bin/systemctl status APPlication
```

Since there is no internet access on the machine where we plan to run Podman, we will need to install dependencies. The versions may be different at the moment, but the set of dependencies should be as follows:

```
attrs-21.4.0.tar.gz
black-22.3.0.tar.gz
build-0.7.0.tar.gz
click-8.0.4.tar.gz
coverage-6.2.tar.gz
dataclasses-0.8.tar.gz
distlib-0.3.4.zip
execnet-1.9.0.tar.gz
filelock-3.4.1.tar.gz
importlib_metadata-4.8.3.tar.gz
importlib_resources-5.4.0.tar.gz
iniconfig-1.1.1.tar.gz
jaraco.context-4.1.1.tar.gz
jaraco.envs-2.2.0.tar.gz
jaraco.functools-3.4.0.tar.gz
jaraco.packaging-9.0.0.tar.gz
jaraco.path-3.3.1.tar.gz
more-itertools-8.12.0.tar.gz
mypy-0.950.tar.gz
mypy_extensions-0.4.3.tar.gz
packaging-21.3.tar.gz
path-16.2.0.tar.gz
pathspec-0.9.0.tar.gz
pep517-0.12.0.tar.gz
pip-run-8.8.0.tar.gz
platformdirs-2.4.0.tar.gz
pluggy-1.0.0.tar.gz
py-1.11.0.tar.gz
pyparsing-3.0.8.tar.gz
pytest-7.0.1.tar.gz
pytest-black-0.3.12.tar.gz
pytest-cov-3.0.0.tar.gz
pytest-enabler-1.2.1.tar.gz
pytest-forked-1.4.0.tar.gz
pytest-mypy-0.9.1.tar.gz
pytest-xdist-2.5.0.tar.gz
python-dotenv-0.20.0.tar.gz
rst.linker-2.2.0.tar.gz
setuptools-40.8.0.zip
setuptools-62.1.0.tar.gz
singledispatch-3.7.0.tar.gz
six-1.16.0.tar.gz
Sphinx-4.5.0.tar.gz
toml-0.10.2.tar.gz
tomli-1.2.3.tar.gz
tomli_w-1.0.0.tar.gz
tox-3.25.0.tar.gz
typed_ast-1.5.3.tar.gz
typing_extensions-4.1.1.tar.gz
virtualenv-20.14.1.tar.gz
wheel-0.37.1.tar.gz
zipp-3.6.0.tar.gz
```

All of the above packages will be needed to run the podman-compose dependencies, which in turn will be as follows:

```
pyenv-2.2.4-1.tar.gz
Python-3.10.2.tar.xz
python-dotenv-0.20.0.tar.gz
PyYAML-5.3.1.tar.gz
setuptools-62.1.0.tar.gz
```

Sample file: requirements.txt

```
[docs]
sphinx
jaraco.packaging>=9
rst.linker>=1.9
```

```
jaraco.tidelift>=1.4
pygments-github-lexers==0.0.5
sphinx-favicon
sphinx-inline-tabs
sphinxcontrib-towncrier
furo

[ssl]

[testing]
pytest>=6
pytest-checkdocs>=2.4
pytest-flake8
pytest-enabler>=1.0.1
pytest-perf
mock
flake8-2020
virtualenv>=13.0.0
wheel
pip>=19.1
jaraco.envs>=2.2
pytest-xdist
jaraco.path>=3.2.0
build[virtualenv]
filelock>=3.4.0
pip_run>=8.8
ini2toml[lite]>=0.9
tomli-w>=1.0.0

[testing-integration]
pytest
pytest-xdist
pytest-enabler
virtualenv>=13.0.0
tomli
wheel
jaraco.path>=3.2.0
jaraco.envs>=2.2
build[virtualenv]
filelock>=3.4.0

[testing:platform_python_implementation != "PyPy"]
pytest-black>=0.3.7
pytest-cov
pytest-mypy>=0.9.1
```

The podman-compose.py itself can be downloaded from GitHub (https://github.com/containers/podman-compose). I also recommend checking out the official Podman instructions and guidelines (https://docs.podman.io/en/latest/index.html). I’ve used v1.0.4 which is currently not available on Github, but there are v0.1.5, and newer versions.

If you have root access to the target virtual machine it's much easier, in my case I had to interact with sudo through a separate engineer, that's Fintech after all.

In my case, the first launch of the system took a lot of time due to the bureaucracy and complexity of interacting with the end machine, but we are building a modern CI/CD, so further updates to the application should be quick and easy.

His Majesty Bash!

Fintech...

It is forbidden to use anything other than bash, moreover, the user we connect to the VM does not have access rights to the home directory of the user Podman, let's start building our monster:

Declaring the script to stop in case of an error in any of the commands.

I also like to decorate the output in stdout with color.

For convenience of further use, let's declare some variables.

$APP_NAME and $API_APP_NAME are declared in CI\CD system (for me it was Teamcity, I wrote the code for Teamcity in Kotlin DSL, but this is a topic for a separate article).

```
#!/bin/bash
### Exit from the script immediately if a command exits with a non-zero status.
set -e

### Decoration by color
PURPLE='\033[0;35m'
NoColor='\033[0m'

### Declare some variables
prometheus_image='xxx/prometheus:2.24.1-ubi8'
fluent_image='fluent/fluent-bit:1.6-debug'
nginx_image='nginx:1.18'

app_targz="$APP_NAME.tar.gz"
api_targz="$API_APP_NAME.tar.gz"
prometheus_targz='prometheus.tar.gz'
fluent_targz='fluent-bit.tar.gz'
nginx_targz='nginx.tar.gz'
```

Next, we need to declare a TRAP to remove the password from the runner and the application image in case the script exits by mistake:

```
##########################################################################################
#########
###                                                                   ###
###                 trap_cleanup function accepts 1 argument                  ###
###                        1st path to target file                         ###
###           Example: trap_cleanup "trap_cleanup $CONTOUR/vault_pass"              ###
###                                                                   ###
##########################################################################################
#########
function trap_cleanup {
  echo -e "\n${PURPLE} TRAP function has been executed! Deleting $1 file${NoColor}"
    chmod 700 "$1"
    rm -rf "$1"
  echo -e "${PURPLE} $1 file successfully deleted.${NoColor}"
}


if [ $CONTOUR = "dev" ]; then
  trap 'trap_cleanup "$CONTOUR/vault_pass"; trap_cleanup "$CONTOUR/vault_pass_local"' EXIT
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  trap 'trap_cleanup "$CONTOUR/vault_pass"; trap_cleanup "$CONTOUR/vault_pass_local"; trap_cleanup
"$CONTOUR/local_pass"' EXIT
```

```
fi
```

For such an important system there were of course more loops, testing, and such, but for simplicity's sake, I left just a few.

My system uses several levels of security and encryption, and since we need to perform all our operations on a remote machine, we need to create files for decryption and remote access, the very files that our traps trigger:

```
### Create a password for SSH and docker to login
if [ $CONTOUR = "dev" ]; then
  echo -e "\n${PURPLE}Create a password for and ssh docker to login...${NoColor}"
  echo "$NEXUS_CI_PASS" > "$CONTOUR/vault_pass"
  chmod 0400 "$CONTOUR/vault_pass"
  echo -e "${PURPLE}Create a password for and ssh docker to login operation completed!${NoColor}\n"
fi

### Create a local password for Ansible vault
echo -e "\n${PURPLE}Create local password for ansible...${NoColor}"
echo "$vault_pass" > "$CONTOUR/vault_pass_local"
chmod 0400 "$CONTOUR/vault_pass_local"
echo -e "${PURPLE}Create local password for ansible! Operation completed!${NoColor}\n"

### Create a local password for SSH on ift and higher
if [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  echo "$VM_LOCAL_PASSWD" > "$CONTOUR/local_pass"
fi
```

Simple decrypt ansible vault, check that the specified file is present and decrypt, or output an error:

```
##########################################################################################
#########
###                                                                      ###
###                        decrypt_file function accepts 1 argument                        ###
###          and will always use "$CONTOUR/vault_pass_local" to decrypt provided file.          ###
###                              1st path to encrypted file                              ###
###                   Example: decrypt_file "$CONTOUR/xxx/keyvault.jks"                   ###
###                                                                      ###
##########################################################################################
#########
function decrypt_file {
  echo -e "\n${PURPLE}Decrypt $1 ${NoColor}"
  test -f "$1" && ansible-vault decrypt "$1" --vault-password-file "$CONTOUR/vault_pass_local" && \
  echo -e "${PURPLE}Decrypt $1 Completed successfully!${NoColor}\n" || \
  echo -e "${PURPLE} Failed to decrypt $1. File not exist or not enough permissions.${NoColor}\n"
}
```

Since the end machine does not have access to our enterprise Docker registry, we need to prepare an image on the runner, download it and archive it:

```
##########################################################################################
#########
###                                                                      ###
###                     prepare_image function accepts 2 or 4 arguments.                     ###
###                    function downloads the image from the repository
###
###                              and archives it to tar.gz
###
###                                 1st - docker repository                                  ###
###                               2nd - application image name                               ###
###                                 3rd - name of tar gzip                                   ###
###                              4th - application tag (version)                             ###
###                                                                                          ###
#########################################################################################
#########
function prepare_image {
  if [ -z "$4" ]; then
    echo -e "\n${PURPLE}Downloading $1/$2 image to agent and saving it to tar.gz${NoColor}"
    docker pull "$1/$2"
    echo -e "${PURPLE} $1/$2 image has been downloaded.${NoColor}"
    echo -e "${PURPLE} Start archiving process for $1/$2 image.${NoColor}"
    docker save "$1/$2" | gzip > "$3"
    echo -e "${PURPLE}Downloading $1/$2 image to agent and saving it to tar.gz. Completed
successfully!${NoColor}\n"
  else
    echo -e "\n${PURPLE}Downloading $1/$2:$4 image to agent and saving it to tar.gz${NoColor}"
    docker pull "$1/$2:$4"
    echo -e "${PURPLE} $1/$2:$4 image has been downloaded.${NoColor}"
    echo -e "${PURPLE} Start archiving process for $1/$2:$3 image.${NoColor}"
    docker save "$1/$2:$4" | gzip > "$3"
    echo -e "${PURPLE}Downloading $1/$2:$4 image to agent and saving it to tar.gz. Completed
successfully!${NoColor}\n"
  fi
}
```

We have to remember to clean up the information on the runner:

```
    #########################################################################################
############
###                                                                                          ###
###                    docker_logout_rmi function accepts 2 or 3 arguments.                  ###
###                function logout agent from docker repo and deletes local image            ###
###                                 1st - docker repository                                  ###
###                               2nd - application image name                               ###
###                              3rd - application tag (version)                             ###
###                                                                                          ###
#########################################################################################
#########
function docker_logout_rmi {
  if [ -z "$3" ]; then
    echo -e "\n${PURPLE}Logout from Nexus $1, to delete credentials from docker config file.${NoColor}"
    docker logout "$1"
    echo -e "${PURPLE}Logout from Nexus $1 Completed successfully!${NoColor}\n"
    echo -e "\n${PURPLE}Delete image $2 from agent.${NoColor}"
    docker rmi "$1/$2"
    echo -e "${PURPLE}Delete image $2 from agent. Completed successfully!${NoColor}\n"
  else
    echo -e "\n${PURPLE}Logout from Nexus $1, to delete credentials from docker config file.${NoColor}"
    docker logout "$1"
    echo -e "${PURPLE}Logout from Nexus $1 Completed successfully!${NoColor}\n"
    echo -e "\n${PURPLE}Delete image $2:$3 from agent.${NoColor}"
    docker rmi "$1/$2:$3"
    echo -e "${PURPLE}Delete image $2:$3 from the agent. Completed successfully!${NoColor}\n"
  fi
}
```

Finally you can transfer the image to the final VM:

```
##########################################################################################
#########
###                                                                      ###
###                  transfer_file function accepts 3 arguments.                   ###
###                   function transfers image to remote host                     ###
###                         1st - file name or path                          ###
###                          2nd - vault pass file                          ###
###                      3rd - VM username for transfer                      ###
###                                                                      ###
##########################################################################################
#########
function transfer_file {
  echo -e "\n${PURPLE}Moving $1 to target host.${NoColor}"
  test -f "$1" && sshpass -f "$2" scp "$1" "$3@$PODMAN_VM:/tmp/" && \
  echo -e "${PURPLE}Moving $1 image to target host. Completed!${NoColor}\n" \
   || \
  echo -e "${PURPLE} Failed to transfer $1. File does not exist or not enough permissions.${NoColor}\n"
}
```

We have transferred the main application, now we need to transfer all the auxiliary images and files:

```
##########################################################################################
#########
###                                                                      ###
###                  transfer_dir function accepts 3 arguments.                    ###
###                   function transfers image to remote host                     ###
###                        1st - Directory name or path                         ###
###                          2nd - vault pass file                          ###
###                      3rd - VM username for transfer                      ###
###                                                                      ###
##########################################################################################
#########
function transfer_dir {
  if test -d "$1"; then
    echo -e "\n${PURPLE}Moving $1 to target host.${NoColor}"
    sshpass -f "$2" scp -r "$1" "$3@$PODMAN_VM:/tmp/" && \
    echo -e "${PURPLE}Moving $1 image to target host. Completed!${NoColor}\n"
  else
    echo -e "${PURPLE} Failed to transfer $1. Directory does not exist or not enough permissions.${NoColor}\n"
    exit 1
  fi
}
```

Let's load the Docker image on the target VM.

As you can see we have to pass commands to run on the remote VM because the podman user does not have login privileges.

```
################################################################################################
#########
###                                                                                      ###
###                          load_image function accepts 3 arguments.                          ###
###                          function transfers image to remote host                          ###
###                               1 st- image archive name or path                          ###
###                                    2nd - vault pass file                                  ###
###                           3rd - VM username for command execution                          ###
###                                                                                      ###
################################################################################################
#########
function load_image {
  echo -e "\n${PURPLE}Load $1 docker image on remote host.${NoColor}"
  sshpass -f "$2" ssh "$3@$PODMAN_VM" "
      sudo -u podman /bin/bash -c 'cd /tmp && \
      podman load -i /tmp/$1'"
  echo -e "${PURPLE}Load $1 docker image on remote host. Completed successfully!${NoColor}\n"
}
```

Let's not forget to clear the home directory:

```
################################################################################################
#########
###                                                                                      ###
###                    clean_up_podman_home function accepts 2 arguments.                    ###
###                              function deletes files and folders                          ###
###             from previous installation in podman home directory on remote host             ###
###                                  1st - vault pass file                                  ###
###                          2nd - VM username for command execution                          ###
###                                                                                      ###
################################################################################################
#########
function clean_up_podman_home {
  echo -e "\n${PURPLE}Cleanup podman home directory on remote host.${NoColor}"
  sshpass -f "$1" ssh "$2@$PODMAN_VM" "
  sudo -u podman /bin/bash -c 'cd /tmp; \
  test -d /app/home/podman/folder1 && chmod -R 755 /app/home/podman/folder1 || echo Failed to chmod folder1 
dir. Dir not exist or not enough permissions; \
  test -f /app/home/podman/folderN/file.env && chmod 755 /app/home/podman/folderN/file.env || echo Failed to 
chmod file.env. File does not exist or not enough permissions; \
  rm -rf /app/home/podman/docker-compose.yaml; \
  rm -rf /app/home/podman/podman_compose.py; \
  rm -rf /app/home/podman/folder1; \
  rm -rf /app/home/podman/folderN'"
  echo -e "\n${PURPLE}Cleanup podman home directory on remote host. Completed 
successfully!${NoColor}\n"
}
```

Now let's move our new release from tmp to the podman home directory:

```
##########################################################################################
#########
###                                                                              ###
###                   move_files_to_podman_home function accepts 2 arguments.                    ###
###                            function moves new files and folders                              ###
###            for new installation from tmp to podman home directory on remote host             ###
###                                  1st - vault pass file                                 ###
###                           2n - VM username for command execution                          ###
###                                                                              ###
##########################################################################################
#########
function move_files_to_podman_home {
  echo -e "\n${PURPLE}Copy all files from tmp to podman home folder.${NoColor}"
  sshpass -f "$1" ssh "$2@$PODMAN_VM" "
    sudo -u podman /bin/bash -c 'cd /tmp && \
    cp -f /tmp/docker-compose.yaml /app/home/podman/docker-compose.yaml && \
    cp -f /tmp/podman_compose.py /app/home/podman/podman_compose.py && \
    cp -rf /tmp/folder1 /app/home/podman/folder1 && \
    cp -rf /tmp/fluent-bit /app/home/podman/fluent-bit && \
    cp -rf /tmp/prometheus /app/home/podman/prometheus && \
    cp -rf /tmp/nginx /app/home/podman/nginx && \
    cp -rf /tmp/$APP_NAME /app/home/podman/$APP_NAME && \
    cp -rf /tmp/$API_APP_NAME /app/home/podman/$API_APP_NAME'"
    echo -e "\n${PURPLE}Copy all files from tmp to podman home folder. Completed
successfully!${NoColor}\n"
}
```

More cleaning:

```
##########################################################################################
#########
###                                                                              ###
###                        clean_up_tmp function accepts 2 arguments.                        ###
###                       function cleans up tmp folder on remote host                        ###
###                                  1st - vault pass file                                 ###
###                          2nd - VM username for command execution                          ###
###                                                                              ###
##########################################################################################
#########
function clean_up_tmp {
  echo -e "\n${PURPLE}Cleanup tmp folder.${NoColor}"
  sshpass -f "$1" ssh "$2@$PODMAN_VM" "
  rm -rf /tmp/$app_targz && \
  rm -rf /tmp/$api_targz && \
  rm -rf /tmp/$prometheus_targz && \
  rm -rf /tmp/$fluent_targz && \
  rm -rf /tmp/$nginx_targz && \
  rm -rf /tmp/fluent-bit && \
  rm -rf /tmp/prometheus && \
  rm -rf /tmp/nginx && \
  rm -rf /tmp/$APP_NAME && \
  rm -rf /tmp/$API_APP_NAME && \
  rm -rf /tmp/podman_compose.py && \
  rm -rf /tmp/folder1 && \
  rm -rf /tmp/docker-compose.yaml"
  echo -e "${PURPLE}Cleanup tmp folder. Completed successfully!${NoColor}\n"
}
```

Depending on the outline, we need to substitute hostname in the prometheus.yml settings:

```
############################################################################################
#########
###                                                                    ###
###          hostname_in_prometheus_config function accepts 2 arguments.          ###
###     function changes HOSTNAME in prometheus.yml based on the host it was deployed.     ###
###                          1st - vault pass file                          ###
###                  2nd - VM username for command execution                  ###
###                                                                    ###
############################################################################################
#########
function hostname_in_prometheus_config {
  echo -e "\n${PURPLE}Change hostname inside prometheus.yml file.${NoColor}"
  sshpass -f "$1" ssh "$2@$PODMAN_VM" "
    sudo -u podman /bin/bash -c 'cd /tmp && \
    sed -i s/@@HOSTNAME@@/${HOSTNAME}/g /app/home/podman/prometheus/prometheus.yml'"
  echo -e "${PURPLE}Change hostname inside prometheus.yml file. Completed successfully!${NoColor}\n"
}
```

Clear the docker cache folder:

```
############################################################################################
#########
###                                                                    ###
###             clean_up_docker_tmp function accepts 2 arguments.             ###
###          function cleans up /var/tmp/docker folder on remote host          ###
###                          1st - vault pass file                          ###
###                  2nd - VM username for command execution                  ###
###                                                                    ###
############################################################################################
#########
function clean_up_docker_tmp {
  echo -e "\n${PURPLE}Clean tmp podman folder on remote host.${NoColor}"
  sshpass -f "$1" ssh "$2@$PODMAN_VM" "
    sudo -u podman /bin/bash -c 'df -h /var && \
    rm -rf /var/tmp/docker* && \
    df -h /var'"
  echo -e "${PURPLE}Clean tmp podman folder on remote host. Completed successfully!${NoColor}\n"
}
```

Let's fix the file permissions:

```
############################################################################################
#########
###                                                                    ###
###               permissions function accepts 2 arguments.               ###
###     function changes secrets.env, xxx folder and xxx files permissions to secure     ###
###                                    1 - vault pass file                                    ###
###                            2 - VM username for command execution                            ###
###                                                                                             ###
#############################################################################################
#########
function permissions {
  echo -e "\n${PURPLE}Change permissions for xxx and secret files.${NoColor}"
  sshpass -f "$1" ssh "$2@$PODMAN_VM" "
    sudo -u podman /bin/bash -c 'cd /tmp && \
    chmod 644 /app/home/podman/folder1/* && \
    chmod 711 /app/home/podman/folder1 && \
    chmod 600 /app/home/podman/folderN/secrets.env'"
  echo -e "\n${PURPLE}Change permissions for folderN and secret files. Completed successfully!${NoColor}\n"
}
```

Finally, we can run podman-compose:

```
#############################################################################################
#########
###                                                                                             ###
###                    execute_podman_compose function accepts 3 arguments.                    ###
###         function changes secrets.env, folder1 folder and folder1 files permissions to secure         ###
###                                   1st - vault pass file                                   ###
###                           2nd - VM username for command execution                           ###
###                                 3rd - path to python binary                                 ###
###                                                                                             ###
#############################################################################################
#########
function execute_podman_compose {
  echo -e "\n${PURPLE}Execute podman_compose.py up -d --force-recreate.${NoColor}"
  sshpass -f "$1" ssh "$2@$PODMAN_VM" "
    sudo -u podman /bin/bash -c 'cd /app/home/podman/ && \
    chmod +x /app/home/podman/podman_compose.py && \
    $3 podman_compose.py up -d --force-recreate'"
  echo -e "${PURPLE}Execute podman_compose.py. Completed successfully!${NoColor}\n"
}
```

All that remains is to run all these functions in the sequence we want:

```
############################################################################################
############################################################################################
############################################################################################



### Decrypt secrets.env
decrypt_file "$CONTOUR/$APP_NAME/secrets.env"



### Decrypt application jks
if [ $CONTOUR = "dev" ]; then
  decrypt_file "$CONTOUR/folder1/application.jks"
  decrypt_file "$CONTOUR/folder1/application.key"
elif [ $CONTOUR = "ift" ]; then
```

```
  ...
fi


### Pull $APP_NAME image and save it to tar gz
if [ $CONTOUR = "dev" ]; then
  prepare_image "$DOCKER_CI" "$APP_NAME" "$app_targz"
  prepare_image "$DOCKER_CDL" "$API_APP_NAME" "$api_targz" "$API_APP_VER"
  prepare_image "$DOCKER_COMMON" "$fluent_image" "$fluent_targz"
  prepare_image "$DOCKER_COMMON" "$prometheus_image" "$prometheus_targz"
  prepare_image "$DOCKER_CDL" "$nginx_image" "$nginx_targz"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  ...
fi


### Logout from Nexus-CI, and cleanup local repo on agent.
if [ $CONTOUR = "dev" ]; then
  docker_logout_rmi "$DOCKER_CI" "$APP_NAME"
  docker_logout_rmi "$DOCKER_CD" "$API_APP_NAME" "$API_APP_VER"
  docker_logout_rmi "$DOCKER_COMMON" "$fluent_image"
  docker_logout_rmi "$DOCKER_COMMON" "$prometheus_image"
  docker_logout_rmi "$DOCKER_CD" "$nginx_image"
elif [ $CONTOUR = "ift" ]; then
  ...
fi


### Move all images to target host
if [ $CONTOUR = "dev" ]; then
  transfer_file "$app_targz" "$CONTOUR"/vault_pass "$NEXUS_CI_USER@domain-name.com"
  transfer_file "$api_targz" "$CONTOUR"/vault_pass "$NEXUS_CI_USER@domain-name.com"
  transfer_file "$fluent_targz" "$CONTOUR"/vault_pass "$NEXUS_CI_USER@domain-name.com"
  transfer_file "$prometheus_targz" "$CONTOUR"/vault_pass "$NEXUS_CI_USER@domain-name.com"
  transfer_file "$nginx_targz" "$CONTOUR"/vault_pass "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  transfer_file "$app_targz" "$CONTOUR"/local_pass "$VM_LOCAL_USER"
  transfer_file "$api_targz" "$CONTOUR"/local_pass "$VM_LOCAL_USER"
  transfer_file "$fluent_targz" "$CONTOUR"/local_pass "$VM_LOCAL_USER"
  transfer_file "$prometheus_targz" "$CONTOUR"/local_pass "$VM_LOCAL_USER"
  transfer_file "$nginx_targz" "$CONTOUR"/local_pass "$VM_LOCAL_USER"
fi



### Move docker-compose, podman-compose and folder1 files to target host
echo -e "\n${PURPLE}Moving docker-compose, podman-compose and folder1 files to target host.${NoColor}"
chmod -R 777 "$CONTOUR"/folder1
if [ $CONTOUR = "dev" ]; then
  transfer_file "$CONTOUR/docker-compose.yaml" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-
name.com"
  transfer_file "podman_compose.py" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
  sshpass -f "$CONTOUR"/vault_pass scp -pr "$CONTOUR"/folder1 "$NEXUS_CI_USER"@domain-
name.com@"$PODMAN_VM":/tmp/folder1
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
```

```
  transfer_file "$CONTOUR/docker-compose.yaml" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  transfer_file "podman_compose.py" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  sshpass -f "$CONTOUR"/local_pass scp -pr "$CONTOUR"/folder1
"$VM_LOCAL_USER"@"$PODMAN_VM":/tmp/folder1
fi
echo -e "${PURPLE}Moving docker-compose, podman-compose and folder1 files to target host. Completed
successfully!${NoColor}\n"


### Move configmap for $API_APP_NAME and $APP_NAME. Move fluent-bit + prometheus + nginx config
files to remote host
if [ $CONTOUR = "dev" ]; then
  transfer_dir "$CONTOUR/$APP_NAME" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-
name.com"
  transfer_dir "$CONTOUR/$API_APP_NAME" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-
name.com"
  transfer_dir "$CONTOUR/fluent-bit" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
  transfer_dir "$CONTOUR/prometheus" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
  transfer_dir "$CONTOUR/nginx" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  transfer_dir "$CONTOUR/$APP_NAME" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  transfer_dir "$CONTOUR/$API_APP_NAME" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  transfer_dir "$CONTOUR/fluent-bit" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  transfer_dir "$CONTOUR/prometheus" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  transfer_dir "$CONTOUR/nginx" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
fi


### Restore privileges on secrets.env file.
echo -e "\n${PURPLE}Restore privileges on secrets.env file.${NoColor}"
if [ $CONTOUR = "dev" ]; then
    sshpass -f "$CONTOUR"/vault_pass ssh "$NEXUS_CI_USER"@domain-name.com@"$PODMAN_VM" "
    chmod 644 /tmp/$APP_NAME/secrets.env"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
    sshpass -f "$CONTOUR"/local_pass ssh "$VM_LOCAL_USER"@"$PODMAN_VM" "
    chmod 644 /tmp/$APP_NAME/secrets.env"
fi
echo -e "${PURPLE}Restore privileges on secrets.env file. Completed successfully!${NoColor}\n"


### Cleanup before copy new files to target dir on remote host
if [ $CONTOUR = "dev" ]; then
  clean_up_podman_home "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  clean_up_podman_home "$CONTOUR/local_pass" "$VM_LOCAL_USER"
fi


### Move files to target dir on remote host
if [ $CONTOUR = "dev" ]; then
  move_files_to_podman_home "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  move_files_to_podman_home "$CONTOUR/local_pass" "$VM_LOCAL_USER"
fi
```

```
### Change permissions for folder1 and secret files
if [ $CONTOUR = "dev" ]; then
  folder1_permissions "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  folder1_permissions "$CONTOUR/local_pass" "$VM_LOCAL_USER"
fi


# Clean tmp podman folder on remote host
if [ $CONTOUR = "dev" ]; then
  clean_up_docker_tmp "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  clean_up_docker_tmp "$CONTOUR/local_pass" "$VM_LOCAL_USER"
fi


### Load docker images on remote host.
if [ $CONTOUR = "dev" ]; then
  load_image "$app_targz" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
  load_image "$api_targz" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
  load_image "$prometheus_targz" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
  load_image "$fluent_targz" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
  load_image "$nginx_targz" "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ]; then
  load_image "$app_targz" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  load_image "$api_targz" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  load_image "$prometheus_targz" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  load_image "$fluent_targz" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
  load_image "$nginx_targz" "$CONTOUR/local_pass" "$VM_LOCAL_USER"
fi


### Change hostname inside prometheus.yml file
if [ $CONTOUR = "dev" ]; then
  hostname_in_prometheus_config "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  hostname_in_prometheus_config "$CONTOUR/local_pass" "$VM_LOCAL_USER"
fi


### Execute podman_compose.py up -d --force-recreate on remote host
if [ $CONTOUR = "dev" ]; then
  echo -e "\n${PURPLE}Execute podman_compose.py up -d --force-recreate.${NoColor}"
    sshpass -f "$CONTOUR"/vault_pass ssh "$NEXUS_CI_USER"@domain-name.com@"$PODMAN_VM" "
    sudo -u podman /bin/bash -c 'cd /app/home/podman/ && \
    chmod +x /app/home/podman/podman_compose.py && \
    ./podman_compose.py up -d --force-recreate'"
  echo -e "${PURPLE}Execute podman_compose.py. Completed successfully!${NoColor}\n"
elif [ $CONTOUR = "ift" ]; then
  execute_podman_compose "$CONTOUR/local_pass" "$VM_LOCAL_USER"
"/app/pyenv/versions/3.10.2/bin/python"
elif [ $CONTOUR = "loadtest" ]; then
  execute_podman_compose "$CONTOUR/local_pass" "$VM_LOCAL_USER"
"/app/home/podman/pyenv/versions/3.10.2/bin/python"
```

```
fi


### Cleanup everywhere
if [ $CONTOUR = "dev" ]; then
  clean_up_tmp "$CONTOUR/vault_pass" "$NEXUS_CI_USER@domain-name.com"
elif [ $CONTOUR = "ift" ] || [ $CONTOUR = "loadtest" ]; then
  clean_up_tmp "$CONTOUR/local_pass" "$VM_LOCAL_USER"
fi
```

Conclusion.

So our Podman DMZ CI\CD on BASH is ready, Fintech is happy, security department guys are happy too, normal people are shocked and went to take pills.

In general, configuring and debugging Podman is a complicated procedure, Podman-compose is even more complicated, and Podman-compose in DMZ without access to VM is almost impossible, but as you can see it is solvable.

Honestly speaking debugging and configuration of VM, Teamcity, Podman and the whole pipeline as a whole took a lot of sleep for me, at some moments podman simply refused to start containers or crashed for unknown reasons, and neither Google nor Stackoverflow did not help.

In conclusion, the integration of Podman with BASH CI/CD pipelines offers numerous benefits. It provides a secure, efficient, and automated way to manage the software development process. As such, it is a valuable tool for any development team looking to improve their workflow and increase their productivity in a secure environment.

Future work could explore more advanced uses of these tools, such as integrating them with other technologies or tools or using them to automate more complex tasks. Regardless, the potential of Podman and BASH in CI/CD technology is clear, and we look forward to seeing how they will continue to shape the future of software development.

References / Список литературы

1. Official Podman project website. [Electronic Resource]. URL: https://podman.io (date of access: 15.11.2023).
2. Official GNU Bash project website. [Electronic Resource]. URL: https://www.gnu.org/software/bash (date of access: 05.12.2023).
3. Official podman-compose Github registry. [Electronic Resource]. URL: https://github.com/containers/podman-compose (date of access: 30.11.2023).