

DevOps

Componentes do Grupo

Nome: Flavio Sousa Vasconcelos **RM552421**

Nome: João Carlos França Figueiredo **RM 97887**

Nome: Leonardo Oliveira Esparza **RM550200**

Nome: Wellington Urcino **RM552368**

Virtualização e Docker no Azure DevOps para Governança de Dados do LinkedIn

No ambiente dinâmico do desenvolvimento de software contemporâneo, a integração contínua e a entrega contínua (CI/CD) são pedras angulares para assegurar a eficácia e a confiabilidade dos projetos de software. O DevOps emerge como uma filosofia facilitadora da colaboração entre equipes de desenvolvimento e operações, automatizando processos e elevando a qualidade do software entregue. Dentro dessa esfera, plataformas como o Azure DevOps e tecnologias de virtualização, notadamente o Docker, desempenham funções vitais.

O Azure DevOps, uma plataforma de serviços oferecida pela Microsoft, apresenta um conjunto abrangente de ferramentas para gerenciamento de código-fonte, compilação, testes, implantação e monitoramento de aplicativos. Uma de suas características proeminentes é a integração fluida com serviços em nuvem, proporcionando flexibilidade e escalabilidade para equipes de desenvolvimento. Dentro do Azure DevOps, a virtualização é essencial na criação de ambientes isolados e replicáveis para testes e implantação de aplicativos.

Ao empregar uma Máquina Virtual (VM) no Azure DevOps, os desenvolvedores têm a capacidade de estabelecer um ambiente de desenvolvimento consistente e controlado. Isso é crucial para garantir a reprodução de bugs e testes em um ambiente semelhante ao de produção, minimizando discrepâncias entre ambientes. Além disso, a virtualização possibilita escalabilidade sob demanda, permitindo que as equipes provisionem recursos conforme necessário, sem os encargos e a complexidade associados à infraestrutura física.

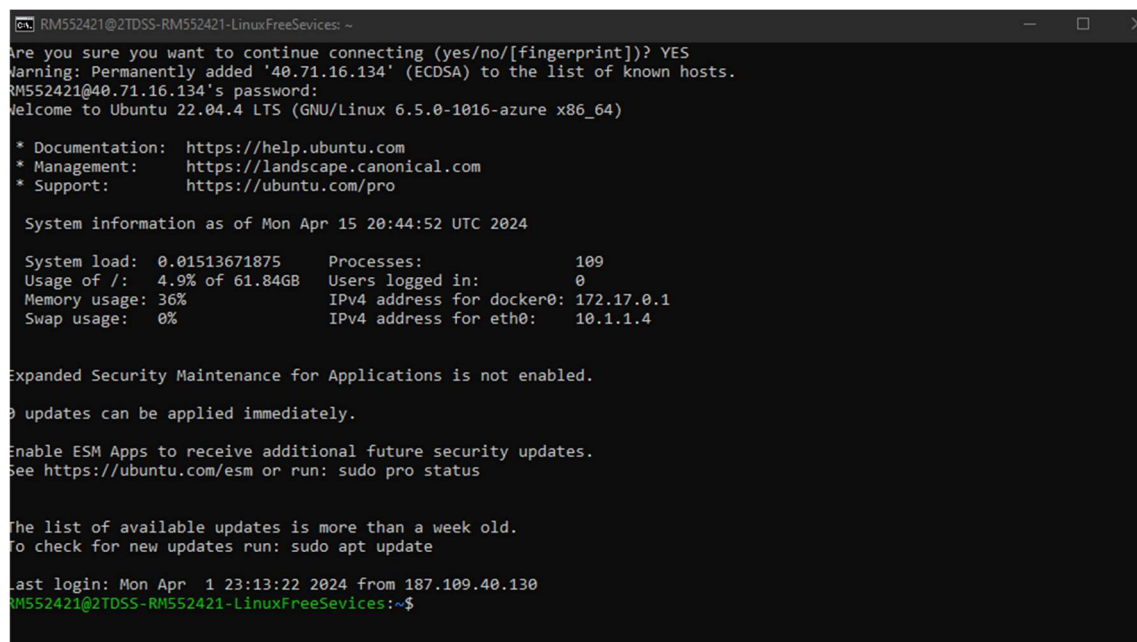
Entretanto, a segurança dos dados figura como uma preocupação preponderante em qualquer ambiente de computação em nuvem. É nesse ponto que o Docker se destaca como uma ferramenta de virtualização de contêineres leve e segura. Ao utilizar contêineres Docker dentro da VM no Azure DevOps, os desenvolvedores

podem garantir uma camada adicional de isolamento e segurança para seus aplicativos e dados. Os contêineres Docker encapsulam o aplicativo e suas dependências em um ambiente controlado, assegurando que qualquer interação com dados externos seja realizada de maneira segura e controlada.

No contexto do projeto mencionado, a utilização de um contêiner Docker para a captação de dados do LinkedIn oferece diversas vantagens. Em primeiro lugar, ao manter controle total sobre o ambiente do contêiner dentro da VM no Azure DevOps, os desenvolvedores podem assegurar que os dados sejam manipulados em conformidade com as políticas de segurança da organização. Em segundo lugar, ao criar um banco de dados SQL (Oracle) dentro do contêiner, os desenvolvedores podem centralizar e gerenciar os dados de forma eficiente, facilitando análises e processamentos posteriores.

Dado as informações acima, a partir de uma máquina virtual será necessário criar um container utilizando Docker na máquina virtual:

Conectando a máquina:

A terminal window titled 'RM552421@2TDSS-RM552421-LinuxFreeSevices: ~' showing the login process for an Ubuntu 22.04.4 LTS system. The user is prompted to continue connecting, and the system displays various status messages and system information. The terminal output includes a warning about adding a host to the list of known hosts, a password prompt, and a welcome message. It then lists documentation, management, and support links. System information is displayed, including system load, processes, memory usage, and network addresses. A message indicates that expanded security maintenance for applications is not enabled, and a prompt to check for updates is shown. The terminal ends with the last login information and the user's shell prompt.

```
RM552421@2TDSS-RM552421-LinuxFreeSevices: ~
Are you sure you want to continue connecting (yes/no/[fingerprint])? YES
Warning: Permanently added '40.71.16.134' (ECDSA) to the list of known hosts.
RM552421@40.71.16.134's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1016-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Mon Apr 15 20:44:52 UTC 2024

System load:  0.01513671875   Processes:            109
Usage of /:   4.9% of 61.84GB   Users logged in:      0
Memory usage: 36%             IPv4 address for docker0: 172.17.0.1
Swap usage:   0%               IPv4 address for eth0:   10.1.1.4

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Apr  1 23:13:22 2024 from 187.109.40.130
RM552421@2TDSS-RM552421-LinuxFreeSevices:~$
```

Upgrade do Linux

```
RM552421@2TDSS-RM552421-LinuxFreeServices: ~
Processing triggers for install-info (6.8-4build1) ...
Processing triggers for linux-image-6.5.0-1018-azure (6.5.0-1018.19~22.04.2) ...
/etc/kernel/postinst.d/initramfs-tools:
update-initramfs: Generating /boot/initrd.img-6.5.0-1018-azure
/etc/kernel/postinst.d/zz-update-grub:
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/40-force-partuuid.cfg'
Sourcing file `/etc/default/grub.d/50-cloudimg-settings.cfg'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
GRUB_FORCE_PARTUUID is set, will attempt initrdless boot
Found linux image: /boot/vmlinuz-6.5.0-1018-azure
Found initrd image: /boot/initrd.img-6.5.0-1018-azure
Found linux image: /boot/vmlinuz-6.5.0-1016-azure
Found initrd image: /boot/initrd.img-6.5.0-1016-azure
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
Scanning processes...
Scanning candidates...
Scanning linux images...

Restarting services...
/etc/needrestart/restart.d/systemd-manager
systemctl restart chrony.service packagekit.service polkit.service rsyslog.service serial-getty@ttyS0.service systemd-journald.service systemd-networkd.service systemd-resolved.service systemd-udevd.service walinuxagent.service
Service restarts being deferred:
systemctl restart getty@tty1.service
systemctl restart networkd-dispatcher.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service
systemctl restart user@1000.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
RM552421@2TDSS-RM552421-LinuxFreeServices:~$
```

Instalação Docker

```
RM552421@2TDSS-RM552421-LinuxFreeServices: ~
systemctl restart networkd-dispatcher.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service
systemctl restart user@1000.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
RM552421@2TDSS-RM552421-LinuxFreeServices:~$ sudo apt-get install
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
RM552421@2TDSS-RM552421-LinuxFreeServices:~$ curl apt-transport-https ca-certificates software-properties-common
curl: (6) Could not resolve host: apt-transport-https
curl: (6) Could not resolve host: ca-certificates
curl: (6) Could not resolve host: software-properties-common
RM552421@2TDSS-RM552421-LinuxFreeServices:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add-
Usage: apt-key [--keyring file] [command] [arguments]

Manage apt's list of trusted keys

  apt-key add <file>      - add the key contained in <file> ('-' for stdin)
  apt-key del <keyid>     - remove the key <keyid>
  apt-key export <keyid>  - output the key <keyid>
  apt-key exportall       - output all trusted keys
  apt-key update          - update keys using the keyring package
  apt-key net-update      - update keys using the network
  apt-key list            - list keys
  apt-key finger          - list fingerprints
  apt-key adv             - pass advanced options to gpg (download key)

If no specific keyring file is given the command applies to all keyring files.
RM552421@2TDSS-RM552421-LinuxFreeServices:~$ sudo apt-get install
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
RM552421@2TDSS-RM552421-LinuxFreeServices:~$
```

```
RM552421@2TDSS-RM552421-LinuxFreeServices: ~
5:23.0.4-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.3-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.2-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.1-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.0-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.24~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.23~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.22~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.21~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.20~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.19~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.18~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.17~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.16~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.15~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.14~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.13~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
RM552421@2TDSS-RM552421-LinuxFreeServices:~$ sudo apt install docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker-ce is already the newest version (5:26.0.1-1~ubuntu.22.04~jammy).
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
RM552421@2TDSS-RM552421-LinuxFreeServices:~$
```

```
RM552421@2TDSS-RM552421-LinuxFreeServices: ~
import      Import the contents from a tarball to create a filesystem image
inspect     Return low-level information on Docker objects
kill         Kill one or more running containers
load        Load an image from a tar archive or STDIN
logs        Fetch the logs of a container
pause       Pause all processes within one or more containers
port        List port mappings or a specific mapping for the container
rename      Rename a container
restart     Restart one or more containers
rm          Remove one or more containers
rmi         Remove one or more images
save        Save one or more images to a tar archive (streamed to STDOUT by default)
start       Start one or more stopped containers
stats       Display a live stream of container(s) resource usage statistics
stop        Stop one or more running containers
tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top         Display the running processes of a container
unpause     Unpause all processes within one or more containers
update      Update configuration of one or more containers
wait        Block until one or more containers stop, then print their exit codes

Global Options:
--config string      Location of client config files (default "/root/.docker")
-c, --context string  Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context set with "docker context use")
-D, --debug          Enable debug mode
-H, --host list       Daemon socket to connect to
-l, --log-level string Set the logging level ("debug", "info", "warn", "error", "fatal") (default "info")
--tls               Use TLS; implied by --tlsverify
--tlscacert string   Trust certs signed only by this CA (default "/root/.docker/ca.pem")
--tlscert string     Path to TLS certificate file (default "/root/.docker/cert.pem")
--tlskey string      Path to TLS key file (default "/root/.docker/key.pem")
--tlsverify          Use TLS and verify the remote
-v, --version        Print version information and quit

Run 'docker COMMAND --help' for more information on a command.

For more help on how to use Docker, head to https://docs.docker.com/go/guides/
RM552421@2TDSS-RM552421-LinuxFreeServices:~$ docker -v
Docker version 26.0.1, build d260a54
RM552421@2TDSS-RM552421-LinuxFreeServices:~$
```

Após isso criaremos o container onde serão os dados serão disponibilizados para captura através do SQL.

Conclusão, a combinação de virtualização no Azure DevOps e o uso de contêineres Docker oferece uma abordagem robusta e segura para o desenvolvimento e implantação de aplicativos na nuvem. Essas tecnologias não apenas aumentam a eficiência e a escalabilidade dos processos de desenvolvimento, mas também garantem a segurança e a governança dos dados, aspectos essenciais em qualquer ambiente de computação em nuvem. Ao compreender e aplicar esses conceitos, os profissionais de DevOps podem impulsionar a inovação e a qualidade dos projetos de software em suas organizações.