ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

AVALIAÇÃO CONTÍNUA II:
Docker e Trabalho Prático

Alonso Lima Machado
(Nº 8190006)

Trabalho Prático apresentado no âmbito da unidade curricular de Computação Distribuída e Nuvem, 1º ano do Mestrado em Engenharia Informática

Docente: Prof. Doutor Ricardo Costa

2019-2020

ÍNDICE GERAL

1.DOCKER

1.1. O que é;

Docker (https://www.docker.com) é a plataforma de container mais utilizada a nível mundial, no modelo SaaS.

1.1.1. O que é um Container?

Um container é um ambiente isolado que contém tudo o que é necessário para que um qualquer software funcione. Ao contrário das Máquinas Virtuais, os containers não são compostos por um sistema operativo completo – apenas contêm as bibliotecas e as configurações necessárias para que o software em questão funcione. Assim, podemos criar sistemas eficientes, simples e auto-contidos, garantindo que os mesmos irão correr de forma independentemente de onde são deployed.

1.2. Como surgiu;

Surgiu da necessidade de compatibilidade porque aplicações precisavam acesso a versões de bibliotecas diferentes, e assim gerava problemas com deploy da aplicação em produção e dificultava os testes.

1.3. Para que serve;

Os desenvolvedores usam Docker para eliminar os problemas de "trabalhar na minha máquina" quando colaboram no desenvolvimento de código com colegas de trabalho.
Os Administradores de Sistemas usam Docker para executar e gerir aplicações lado a lado em containers isolados para obter uma melhor densidade de computação.
As empresas usam Docker para construir agile software delivery pipelines, agilizando a disponibilização de novas funcionalidades de forma mais rápida, segura e confiável para servidores Linux e Windows.

1.4. Como funciona;

O container é uma camada de aplicação que é executada sobre o OS do Host, ou seja, todas as chamadas de sistemas, I/O, rede são tratados pelo OS do Host do docker, portanto cada “Container” tem suas bibliotecas e aplicações instaladas para conseguir rodar e assim funciona independentemente de outra aplicação, evitando assim problemas de compatibilidade.
Até facilita o update para novas versões de softwares, pois cada um é independente em seu próprio “Container” então atualizar o seu Banco de Dados ou Servidor Web para a versão mais recente e estável com mais features é muito mais simples.
Facilita também a CI/CD,

1.5. Como se diferencia de uma máquina virtual tradicional;

A principal diferença é que em uma virtualização tradicional, usando por exemplo o virtualbox, é necessário alocar um grande espaço em disco (HD) e memória ram para a máquina virtual e assim sobrecarrega a máquina Host que a cada nova Virtualização precisa executar 2, 3 ou mais Sistemas Operacionais (OS) além do próprio, causando um overhead de processamento gigantesco e caso existir problemas e for necessário reiniciar/substituir a máquina o tempo necessário é muito superior pois temos de matar a máquina virtual antiga e instanciar uma nova, esperar o Sistema Operacional carregar e depois disso ela estará pronta para uso, já ao usarmos Containers como ele é usualmente bem leve em poucos segundos se instancia um novo porque na realidade basicamente estamos só abrindo uma aplicação nova no nosso Host com algumas bibliotecas específicas ele é um SaaS,Software-as-a-Service,

1.6. Flexibilidade e escalabilidade;

O uso de Containers deixa o ambiente altamente escalável e flexível, facilitando o crescimento do serviço/aplicação.

1.7. Alternativas;

LXC com LXD
Mesos (Apache)

1.8. Como integra com as soluções Cloud existentes (e.g.: Azure, AWS, GCP);

Containers Dockers estão sendo bastante adotados na indústria, o termo Container-as-a-Service está se difundindo.
Existem várias opções:
GCE Google Container Engine - Google Cloud Plataform- Google
ECS EC2 Container Service - AWS - Amazon
PKS Pivotal Container Service - Pivotal CF - Pivotal
Docker Swarm - Docker Cloud - Docker

1.9. Como interage com outras ferramentas (e.g.: Terraform e Kubernetes).

Kubernetes é um Orquestrador de Containers, assim como OpenShift, ou seja, eles basicamente são uma camada de abstração acima do docker para facilitar o gerenciamento e automatização de containers (Rede,ReplicaSets,Deployments,Ingress…).

Vagrant auxilia a criar VMs com imagens e aplicações padronizadas, ótimo para testes ou ao contratar novos desenvolvedores.
Jenkins é uma ferramenta de Build para CI.
Robot Framework Testes nessa Build do Jenkins, e está completo o CI.

2. Trabalho Prático 2 MEI 2019/2020 ESTG.

2.1 Considerações sobre o Trabalho Prático 2

Realmente este trabalho foi mais complicado do que eu esperava, pois já tinha feito o primeiro trabalho prático em RMI, mas neste sofri algumas complicações quanto a volume, rede, dockerfile e execução automática.

2.2 Passo a Passo

Execução por linhas de comando em docker, passo a passo:

2.2.1 Passo 1 - Rede e Docker Builds:

docker network create --driver bridge cdnt2alonso (Cria a rede bridge customizada)
docker image build . -t servidorrmi (Cria a imagem pelo Dockerfile do servidor)
docker image build . -t clientermi (Cria a imagem pelo Dockerfile do cliente)

2.2.2 Passo 2 - Executar os Containers:

2.2.2.1 Servidor:

docker run -dit -p 1099:1099 --name servidor --network cdnt2alonso -v servervol:/serverdata servidorrmi ash

2.2.2.2 Cliente:

docker run -dit --name cliente --network cdnt2alonso -v clientvol:/clientdata clientermi ash

2.2.2.2 Cliente Extra Opcional:

docker run -it --name cliente2 --network cdnt2alonso -v clientvol:/clientdata clientermi ash (Este em modo atrelado ao seu prompt) -it

2.2.3 Passo 3 - Entrar na shell do Servidor:

docker exec -it servidor ash
docker exec -it cliente ash

2.2.1 Passo 4 (Extra) - Publicar no DockerHub

2.2.1.1.Servidor:

docker tag servidorrmi alonsomachado/arquivocdn:servidorrmi
docker push alonsomachado/arquivocdn:servidorrmi

2.2.1.1.Cliente:

docker tag clientermi alonsomachado/arquivocdn:clientermi
docker push alonsomachado/arquivocdn:clientermi

2.2.1 Passo 5 (Extra) - Publicar um tutorial pelo Readme no dockerhub.

Criei um readme no meu repositório do docker hub:
https://hub.docker.com/repository/docker/alonsomachado/arquivocdn

REFERÊNCIAS

https://www.alura.com.br/artigos/desvendando-o-dockerfile
https://hub.docker.com/repository/docker/alonsomachado/arquivocdn
https://docs.docker.com/network/bridge/
https://docs.docker.com/network/network-tutorial-standalone/
https://docs.docker.com/storage/volumes/
https://docs.docker.com/network/network-tutorial-standalone/
https://www.digitalocean.com/community/tutorials/how-to-share-data-between-docker-containers#step-3-%E2%80%94-creating-a-volume-from-an-existing-directory-with-data

ANEXOS

Anexo A. Dockerfile Servidor

```
FROM alpine
WORKDIR /servidor
RUN apk update
RUN apk fetch openjdk8
RUN apk add openjdk8
ENV JAVA_HOME=/usr/lib/jvm/java-1.8-openjdk
ENV PATH="$JAVA_HOME/bin:${PATH}"
COPY policy.all .
COPY Servidor.java .
COPY ClientMonitor.java .
COPY ServidorImpl.java .
COPY Client.java .
COPY run.sh .
RUN javac *.java
ENTRYPOINT chmod 777 /serverdata
ENTRYPOINT chmod 777 -R /servidor
ENTRYPOINT ./run.sh
```

Anexo B. Dockerfile Cliente

```
FROM alpine
WORKDIR /cliente
RUN apk update
RUN apk fetch openjdk8
RUN apk add openjdk8
ENV JAVA_HOME=/usr/lib/jvm/java-1.8-openjdk
ENV PATH="$JAVA_HOME/bin:${PATH}"
COPY policy.all .
COPY Servidor.java .
COPY ClientMonitor.java .
COPY Client.java .
COPY run.sh .
RUN javac *.java
ENTRYPOINT chmod 777 /clientdata
ENTRYPOINT chmod 777 -R /cliente
ENTRYPOINT ./run.sh
```

Anexo C. Interfaces e Run.sh

Servidor

Interface do Servidor

```
public interface Servidor extends java.rmi.Remote {

//Guarda o arquivo na listaArquivos no servidor em Memoria.
   public void guardaArquivo() //String utilizadorEnviouArquivo,
         throws java.rmi.RemoteException;

//Cria um Utilizador logado na hastable para criar threads
   public void addListener(ClientMonitor clientInterface)
         throws java.rmi.RemoteException;

//Remove um Utilizador logado na hastable para criar threads
   public void removeListener(ClientMonitor clientInterface)
         throws java.rmi.RemoteException;

}
```

ClientMonitor

Interface que tem os Listeners de Callbacks do Cliente

```
public interface ClientMonitor extends java.rmi.Remote {

   // Callback que recebe mensagem do servidor para o cliente a referir que o utilizador
recebeu um arquivo
   public void existeArquivoListeners(String msg, String dataEnvio) throws
java.rmi.RemoteException;

         // Callback que recebe o arquivo do servidor e coloca numa pasta no cliente
   public void receberArquivo(String checksum, java.io.File arquivo, byte[] arquivoData)
throws java.rmi.RemoteException;

}
```

Servidor Run.sh

```
javac *.java
java -Djava.security.policy=policy.all ServidorImpl servidor
```

Cliente Run.sh

```
javac *.java
java -Djava.security.policy=policy.all Client servidor
```

Anexo D. Código Java RMI do Servidor

```
import java.rmi.*;
import java.io.*;
import java.rmi.server.*;
import java.io.FileNotFoundException;
import java.net.MalformedURLException;
import java.rmi.registry.LocateRegistry;
import java.io.File;
import java.io.FileWriter;
import java.math.BigInteger;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ConcurrentModificationException;
import java.util.Random;
import java.util.Vector;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;

/**
 * @author Alonso Lima Machado Para o Segundo Trabalho de Mestrado de
 * ComputaÃ§Ã£o DistribuÃda e em Nuvem do Mestrado em Engenharia InformÃ¡tica
 * da ESTG/IPP 2019
 */
public class ServidorImpl extends UnicastRemoteObject implements Servidor, Runnable {

    private static final long serialVersionUID = 4L;            //Serial version uid
    private static ServidorImpl rmi;
          public static String enderecoServidor = "//localhost:1099/Arquivo"; //Endereco ip
deste servidor
          public static String enderecoServidorDocker = "//servidor:1099/Arquivo"; //Endereco
ip no docker servidor (Melhor pegar em tempo de execucao com argv[0])

    public static final Vector<ClientMonitor> listaUserInterface = new Vector<ClientMonitor>();
//Lista de Utilizadores Online Ativos
    public final Vector<File> listaArquivos = new Vector<File>();

    public ServidorImpl() throws RemoteException {

    }
    //Cria o arquivo no Servidor na pasta serverdata, coloca o arquivo memória e envia
posteriormente
```

```
    @Override
    public void guardaArquivo() {

        String utilizador = Thread.currentThread().getName();
                if(listaUserInterface.isEmpty()==false){ //Caso exista Listener na lista cria
arquivo
                        String nomeRandom = "";
                        String md5 = "";
                        Date data = new Date();
                        DateFormat osLocalizedDateFormat = new
SimpleDateFormat("dd/MM/YYYY HH:mm:ss");
                        int i;
                        Random ra = new Random();
                        do{
                        nomeRandom = nomeRandom + ra.nextInt(999999);
                        }while(nomeRandom.length() < 30);
                        //Boolean pastaCriada = new File("..\\serverdata\\").mkdirs();
                        File arq = new File("/serverdata/" + nomeRandom +".txt");
                        FileWriter arquivoNovo;
                        try {

                                arquivoNovo = new FileWriter(arq);
                                arquivoNovo.write("Arquivo Criado em Java para o Trabalho
de CDN do Mestrado do ESTG! Aluno Alonso Machado! \n");
                                arquivoNovo.write("Trabalho 2 Computacao Distribuida e
Nuvem com Docker e Rede e Volume! \n");
                                arquivoNovo.write(nomeRandom); //Escreve dentro do arquivo
o nome dele totalmente randomico
                                listaArquivos.add(arq);
                                arquivoNovo.close();
                                synchronized(this){
                                        md5 = calculaMD5(arq);
                                        System.out.println("MD5 CHECKSUM: "+ md5);
                                        }
                        } catch (FileNotFoundException ex) {
                                System.out.println("Erro ao criar arquivo randomico: " +
nomeRandom + " Erro: " + ex);

                        } catch (NullPointerException e) {
                                System.out.println("Erro NullPointer ao criar arquivo
randomico: " + e);
                        } catch (IOException ex) {
                                System.out.println("Erro IOException ao criar arquivo
randomico: " + ex);
                        }
```

```
                        //Manda para o Cliente Que está esperando o arquivo

                        for (i = 0; i < listaUserInterface.size(); i++) {
                                if (utilizador == null ? listaUserInterface.get(i).toString() == null
: utilizador.equals(listaUserInterface.get(i).toString())) {
                                        ClientMonitor listenerUtilizadorArq =
listaUserInterface.get(i);

                                        File arquivo = listaArquivos.lastElement(); //Pega o
ultimo arquivo da Lista em memoria
                                        try {
                                                byte[] arquivoData =
Files.readAllBytes(Paths.get("/serverdata/" + arquivo.getName()) );

listenerUtilizadorArq.existeArquivoListeners(md5, osLocalizedDateFormat.format(new
Date()));
                                                listenerUtilizadorArq.receberArquivo(md5,
arquivo, arquivoData); //Tenta enviar ao cliente
                                                listaArquivos.remove(arquivo); //Remove o
ultimo arquivo da Lista em memoria
                                        } catch (IOException e) {
                                                System.out.println("ERRO no Listener Arquivo,
Removido" + e);
                                                removeListener(listenerUtilizadorArq);

                                        } catch (NullPointerException e) {
                                                System.out.println("ERRO Null Pointer Listener
Arquivo, Removido" + e);
                                                removeListener(listenerUtilizadorArq);
                                        }
                                }
                        }
                        System.out.println(" O arquivo: " + arq.getName() + " esta na sua
Pasta SERVERDATA conforme especificacao do Trabalho");
                }
        }

        /**
         * JAVADOC
         */
        @Override
        public void run() {
            //Variavel de Controle para Thread parar de gerar e enviar arquivos;
                        boolean continua=false;
                        if(listaUserInterface.isEmpty()==false){ //True se estiver vazio False caso
tenha elementos
```

```
                        continua=true;
                    }
        while(continua!=false) {
            try {
                Thread.sleep(5000); //Tempo em milisegundos (5000 5 segundos)

            } catch (InterruptedException eInterup) {
                System.out.println(eInterup.getMessage());
                                    //Thread.interrupt();
            } catch (Exception e) {
                //Thread.interrupt(); //Mata a Thread
            }

            guardaArquivo();
                        if(listaUserInterface.isEmpty()==true){ //True se estiver vazio False
caso tenha elementos
                                    continua=false;
                        }

        }
    }

    @Override
    public void addListener(ClientMonitor clientInterface) {
        listaUserInterface.add(clientInterface);
        String clientex = clientInterface.toString();
        try {
            Thread userThread = new Thread(rmi, clientex);
            System.out.println("Startando a thread para o CLIENTE: " + clientex);
            userThread.start();
        } catch (ConcurrentModificationException eConcorrencia) {
            //System.out.println("Erro ao percorrer a Lista Por ter alterado. Erro:" +
eConcorrencia);
        } catch (NullPointerException eNullPointer) {
            //System.out.println("Erro Null Pointer percorrer a Lista Por ter alterado. Erro:" +
eNullPointer);
        }
        //System.out.println("Entrou um UserListener Novo!" + clientInterface);
    }

    @Override
    public void removeListener(ClientMonitor clientInterface) {
        int i;
        String clientex = clientInterface.toString();
        for (i = 0; i < listaUserInterface.size(); i++) {
            if (listaUserInterface.get(i).equals(clientInterface)) {
```

```
            System.out.println("Removeu o Listener: " + clientex);
            listaUserInterface.remove(i);
                        }
                }
                System.out.println("--- Listagem dos Listeners Ativos neste servidor agora: ");
                if(listaUserInterface.isEmpty()==true) System.out.println(" Nenhum Listeners
Ativo no servidor agora! ");
                for (i = 0; i < listaUserInterface.size(); i++) {
                        System.out.println(i+" UserListener Existente: " +
listaUserInterface.get(i).toString() );
                }

    }

    public static String hex(byte[] bytes) {
        BigInteger bi = new BigInteger(1, bytes);
        return String.format("%0" + (bytes.length << 1) + "x", bi);
    }

    public static String calculaMD5(File f) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            InputStream is = null;
            try {
                is = new BufferedInputStream(new FileInputStream(f));
                byte[] buf = new byte[8192];
                for (int nBytes = is.read(buf, 0, buf.length); nBytes > 0; nBytes = is.read(buf, 0,
buf.length)) {
                    md.update(buf, 0, nBytes);
                }
            } catch (IOException ex) {
                if (is != null) {
                    try {
                        is.close();
                    } catch (IOException ex2) {
                    }
                }
            }
            byte[] digest = md.digest();
            return hex(digest);
        } catch (NoSuchAlgorithmException ex) {
            System.out.println("Erro ao Realizar MD5! " + ex);
        }
        return null;
    }
```

```
    public static void main(String args[]) throws Exception {

        System.setSecurityManager(new RMISecurityManager());
        try {
            rmi = new ServidorImpl();

            LocateRegistry.createRegistry(1099);
            System.out.println("Registry criado");

                                if (args.length != 1)
                                throw new RuntimeException("Syntax:" + " ServidorImpl
<hostname>");
                                Naming.rebind("//"+args[0]+":1099/Arquivo", rmi);
                                System.out.println("Servidor Bindando na Porta 1099");
                                System.out.println("Envia Arquivo Randomicamente Criado para cada
cliente conectado a cada 5 Segundos....");
                                System.out.println("Esperando Clientes....");
                                int i = 0;
            while (true) {

                i++;
                if (i > 999) {
                    i = 1;
                }
            }
                        } catch (java.rmi.UnknownHostException uhe) {
            System.out.println("Erro java.rmi.UnknownHostException.");
        } catch (RemoteException re) {
            System.out.println("Erro ao iniciar servico RMI: " + re);
        } catch (MalformedURLException mURLe) {
            System.out.println("Erro ao instaciar na URL //localhost:1099/IRCServer" + mURLe);
        }
    }
}
```

Anexo E. Código Java RMI do Cliente

```
import java.rmi.*;
import java.rmi.Naming;
import java.io.*;
import java.math.BigInteger;
import java.nio.file.Files;
import java.nio.file.Paths;
import javax.rmi.PortableRemoteObject;
import java.rmi.server.UnicastRemoteObject;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;


/**
 *
 * @author Alonso Lima Machado Para o Segundo Trabalho de Mestrado de
 * ComputaÃ§Ã£o DistribuÃda e em Nuvem do Mestrado em Engenharia InformÃ¡tica
 * da ESTG/IPP 2019
 */
public class Client extends UnicastRemoteObject implements ClientMonitor {

private static final long serialVersionUID = 4L;            //Serial version uid

   protected Client() throws RemoteException {
   }

   public static void main(String args[]) throws Exception {

                  if (args.length != 1)
            throw new RuntimeException("Syntax:" + " Client <hostname>");


      //Instancia o Security Manager para o RMI
      if (System.getSecurityManager() == null) {
          System.setSecurityManager(new RMISecurityManager());
      }

      // Criando registry do Servidor
      //Servidor service = (Servidor) Naming.lookup(enderecoServidor);
                  Servidor service = (Servidor)
Naming.lookup("rmi://"+args[0]+":1099/Arquivo");
      // Criando leitor do Terminal
      DataInputStream din = new DataInputStream(System.in);
```

```
        String line;
        //Construir um objeto dele mesmo
        Client cliente = new Client();
        BufferedOutputStream output;
        System.out.println("Executando Cliente! " );
        service.addListener(cliente);
                    int i = 0;
                    int j = 0;
        while (true) {
           for(i=0;i < 999990;i++){
                    for(j=0;j < 9999;j++){
                            if (i > 999980 & j > 9997) {
                                    i = 1;
                                    j = 1;
                            }
                    }
           }
        }
} //Fecha chaves da MAIN
 @Override
   public void existeArquivoListeners(String checksum, String data) {
      System.out.println("\nRecebeu um arquivo " + "(" + data + ") MD5 Recebido: " +
checksum);
   }

@Override
   public void receberArquivo(String md5Recebido, java.io.File recebido, byte[] arquivoData)
{
                    //Boolean pastaCriada = new File("..\\clientdata\\").mkdirs();
        try {

           String checksum = "";
           OutputStream outputStream = new FileOutputStream("/clientdata/" +
recebido.getName());
           outputStream.write(arquivoData);
           byte[] b = Files.readAllBytes(Paths.get("/clientdata/" + recebido.getName()) );
           byte[] hash = MessageDigest.getInstance("MD5").digest(b);
           checksum = Arrays.toString(hash);
           String md5p = hex(hash);
           System.out.println("HASH CHECKSUM: " + checksum + " MD5: " + md5p);
           System.out.println((md5p == null ? md5Recebido == null :
md5p.equals(md5Recebido)) ? "CHECKSUM e MD5 do Arquivo CORRETO!" : "Arquivo
Com Erro"); //Verificar Checksum
           outputStream.flush();
           outputStream.close();
```

```
            System.out.println(" O arquivo: " + recebido.getName() + " esta na sua Pasta
CLIENTDATA conforme especificacao do Trabalho");

        } catch (FileNotFoundException ex) {
            System.out.println("Arquivo a ser Recebido teve problemas na Transferencia: " + ex);
        } catch (IOException ex) {
            System.out.println("Arquivo Recebido com erro IO: " + ex);
        } catch (NullPointerException e) {
            //System.out.println("Arquivo Recebido deu Null Pointer com erro " + e);
        } catch (NoSuchAlgorithmException ex) { //Nunca ocorre para MD5 mas e obrigatorio
para tirar o warning/error no Netbeans
            System.out.println("Erro ao Realizar MD5! " + ex);
        }
  }

    public static String hex(byte[] bytes) {
        BigInteger bi = new BigInteger(1, bytes);
        return String.format("%0" + (bytes.length << 1) + "x", bi);
    }
}
```