Antes de Apresentar:

- Ligar as 4 VMS

- Ligar a VM-centOS com o EdgeFog

- Ligar a primeira VM com o EventProcessing

- Ligar todas as outras VMs com o FrontEnd

- Ligar o IntelIJ meter o endereço da VM do EdgeFog para o IoTProducer

- Meter a gerar eventos até ter 10 eventos guardados no FrontEnd

- Realizar todas as operações do contrato

DIZER (MADORNA):

- Inicialmente ligamos 4 VMs, onde numa VM temos o RabbitMQ a funcionar num container

- Nas restantes 3 temos todas elas a correr um FrontEnd

- Numa destas 3 VMs temos também um consumidor a correr

- Para facilitar a demonstração temos já 10 eventos guardados no FrontEnd

MOSTRAR (TÁVORA):

* Prints na consola de um dos FrontEnds com os eventos guardados

FAZER (TÁVORA):

* Ligar o IntelIJ para ligar o User e meter o endereço IP de um dos servidores FrontEnd e realizar todas as operações

DIZER (MADORNA):

* Temos disponíveis 5 operações, onde temos a primeira que obtém o valor mais alto, o identificador do dispositivo que obteve esse valor, o local e data de aquisição
* A segunda obtém o valor medio de todas as velocidades
* A Terceira obtém a velocidade media entre duas datas
* O número de consumidores
* Uma simulação de inicialização de um consumidor e exibe os nomes dos consumidores ativos, caso o nome escrito no User seja admin

FAZER (TÁVORA):

* Ligar novo consumidor numa outra VM

DIZER (MADORNA):

* Vamos iniciar um novo consumidor, o grupo de consumidores possui um líder que é responsável por enviar a entrada ou saída de consumidor do EventProcessing

FAZER (TÁVORA):

* Voltar a executar o User e realizar obter numero de consumidores e nomes dos consumidores

DIZER (MADORNA):

* Como se pode observar o User obteve neste caso 2 consumidores e os nomes deles

FAZER (TÁVORA E MADORNA):

* Enviar uma mensagem com valor superior e inferior a 120

MOSTRAR (TÁVORA):

* Mostrar os prints na consola dos FrontEnd e do EdgeFog

DIZER (MADORNA):

* Como se pode observar tanto o FrontEnd como o EdgeFog possuem os eventos enviados

MOSTRAR (TÁVORA):

* Slides do algoritmo de eleição do líder

DIZER (TÁVORA):

* Inicialmente o primeiro membro a juntar-se ao grupo é eleito o líder
* O Segundo membro a juntar-se ao grupo envia uma mensagem a perguntar quem é o líder
* Todos os membros que estão no grupo, incluindo o próprio que perguntou quem é o líder, enviam uma mensagem sobre quem é o seu atual líder
* O membro que ainda não sabe quem é o líder envia uma mensagem com um número aleatório para se tornar o líder.
* O líder envia uma mensagem a dizer que é o líder, sobrepondo-se á mensagem enviada pelo membro que não tem líder e desta forma esse membro fica a saber quem é o líder.
* Um membro que não seja o líder mas já saiba quem é o líder do grupo envia uma mensagem a dizer quem é o seu líder atual.
* Todos os membros recebem todas as mensagens e o líder enviado pelos membros deve ser igual em todas as mensagens
* Caso isto não se verifique, é removido o líder atual de todos os membros e é eleito novo líder
* Um novo líder é eleito quando todos os membros do grupo enviarem uma mensagem afirmar que são o líder, esta mensagem contém um valor aleatório para se tornarem líder
* O membro que tiver o maior valor é eleito líder.
* Caso seja um membro que não seja o líder saia do grupo, não acontece nada
* Caso seja o líder saia do grupo, todos os elementos removem o seu atual líder e enviam uma mensagem com o valor aleatório para se tornarem líder, o valor mais alto torna-se o líder.

FAZER (TÁVORA):

* Terminar um FrontEnd onde não é líder
* Voltar a Ligar um FrontEnd

DIZER (TÁVORA):

* Neste caso como a aplicação que eu terminei não é o líder não é eleito um novo líder e quando se conecta o novo FrontEnd o líder envia uma mensagem em JSON com todos os elementos que possui guardados

FAZER (TÁVORA):

* Terminar o FrontEnd líder
* Voltar a Ligar um FrontEnd

DIZER (TÁVORA):

* Neste caso como a aplicação é líder o grupo elege um novo líder e quando se conecta um novo FrontEnd o novo líder envia a mensagem JSON para o FrontEnd que entrou no grupo.