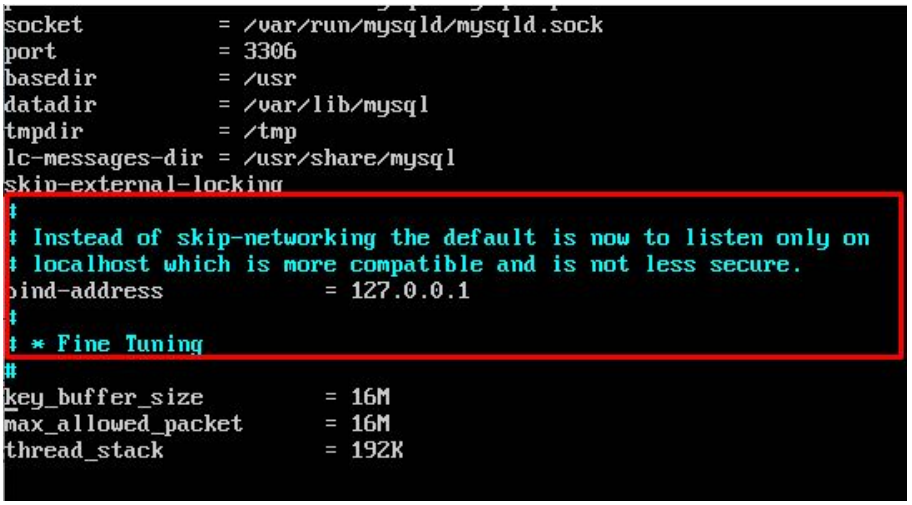
Atenção: Os passos abaixo foram

1. Instalar MySql

Será necessário configurar o banco de dados MySQL para atender como repositório de dados ao sistema Student Lounge. Portanto, recomenda-se instalar a versão mais atual do MySQL no servidor, expondo a melhor porta para acesso externo (Por padrão esta é a 3306).

Sugere-se os processos abaixo para um servidor Ubuntu:

- sudo apt-get update
- sudo apt-get install mysql-server
- sudo mysql_secure_installation utility
- sudo ufw enable
    - Este comando permitirá acesso remoto
- sudo ufw allow mysql
    - Este comando permitirá acesso remoto
- sudo systemctl start mysql
- sudo systemctl enable mysql
- abrir arquivo /etc/mysql/mysql.conf.d/mysqld.cnf em modo edição
- Editar o campo “bind-address” com as configurações abaixo
    - bind-address = 127.0.0.1 (default.)
    - bind-address = 0.0.0.0 (Todo e qualquer endereço.)
    - bind-address = X.X.X.X (Endereço específico. Caso desejado pode manter este e remover o de cima, que dá acesso a todos)



- sudo systemctl restart mysql
- Criar um usuário para acesso externo e/ou da aplicação
    - /usr/bin/mysql -u root -p
    - CREATE USER '<usuario_externo>'@'localhost' IDENTIFIED BY '<senha>';
    - CREATE USER '<usuario_externo>'@'%' IDENTIFIED BY '<senha>';
    - GRANT ALL ON *.* TO '<usuario_externo>'@'localhost';

- GRANT ALL ON *.* TO '<usuario_externo>'@'%';
- FLUSH PRIVILEGES;

2. Configurar aplicação para acesso ao endereço do banco

Toda aplicação executável possui um arquivo de configuração chamado **appsettings** que se encontra na raíz do projeto.

Este arquivo, geralmente, é dividido em dois:

- **appsettings.json**: configurações para produção
- **appsettings.Development.json**: configurações para modo debug. Os parâmetros inexistentes neste, automaticamente herdam de **appsettings.json**.

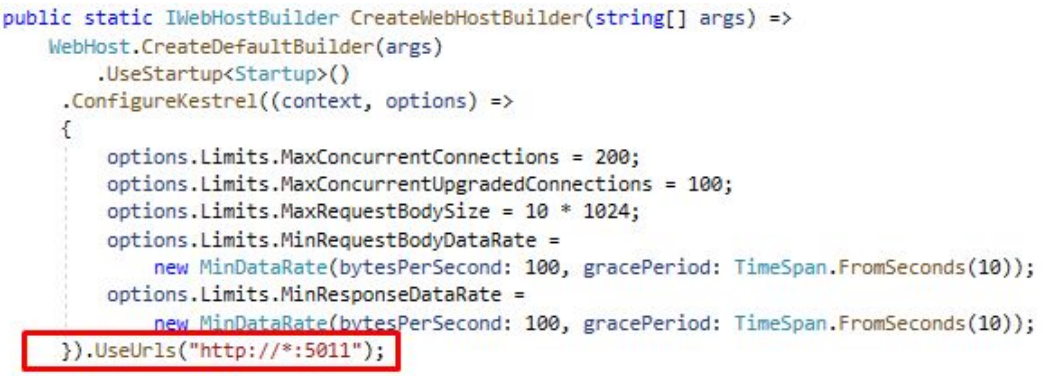
Na seção “ConnectionStrings” é necessário atualizar os dados para que a aplicação possa conectar com o servidor e banco de dados. Caso o o banco esteja no mesmo ambiente onde a aplicação ficará hospedada, pode-se utilizar o endereço “localhost”, caso contrário será necessário utilizar o endereço externo (IP ou DNS) do servidor:

```
"ConnectionStrings": {
  "sLyceumDB": "Server=localhost;uid=USUARIO_BANCODADOS;pwd=SENHA_BANCODADOS;database=SCHEMA_BANCODADOS;",
  "csvDB": "Server=localhost;uid=USUARIO_BANCODADOS;pwd=SENHA_BANCODADOS;database=SCHEMA_BANCODADOS;",
  "StudentLoungeDB": "Server=localhost;uid=USUARIO_BANCODADOS;pwd=SENHA_BANCODADOS;database=SCHEMA_BANCODADOS;",
  "MasterDB": "Server=localhost;uid=USUARIO_BANCODADOS;pwd=SENHA_BANCODADOS;database=SCHEMA_BANCODADOS;"
},
```

Cada parâmetro representa uma conexão com um banco diferente:

- **csvDB:** Conexão para banco de dados/schema responsável por sincronização de carga
- **StudentLoungeDB:** Conexão com banco de dados principal do sistema Student Lounge
- **MasterDB:** Conexão com banco de dados Master do sistema Student Lounge

3. Configurar porta de execução da aplicação
   Quando executada no servidor Linux, a aplicação rodará em uma porta configurada no arquivo **Program.cs** de acordo com a imagem abaixo



```
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
    WebHost.CreateDefaultBuilder(args)
        .UseStartup<Startup>()
     .ConfigureKestrel((context, options) =>
     {
         options.Limits.MaxConcurrentConnections = 200;
         options.Limits.MaxConcurrentUpgradedConnections = 100;
         options.Limits.MaxRequestBodySize = 10 * 1024;
         options.Limits.MinRequestBodyDataRate =
             new MinDataRate(bytesPerSecond: 100, gracePeriod: TimeSpan.FromSeconds(10));
         options.Limits.MinResponseDataRate =
             new MinDataRate(bytesPerSecond: 100, gracePeriod: TimeSpan.FromSeconds(10));
     }).UseUrls("http://*:5011");
```

   Caso seja necessário alterar a porta de execução da aplicação, deverá ser alterada conforme local destacado na figura acima onde se encontra o valor **“5011”**

4. Instalar SDK dotnet Core

   A aplicação utiliza a versão 2.2 da SDK .NET Core, portanto será necessário instalar os pacotes da SDK no ambiente do servidor, seguindo os comandos abaixo:

   - sudo add-apt-repository universe
   - sudo apt-get update
   - sudo apt-get install apt-transport-https
   - sudo apt-get update
   - sudo apt-get install dotnet-sdk-2.2

5. Publicar e armazenar compilado em pasta no servidor
   A aplicação hospedada em um servidor deve ser compilada, portanto será necessária gerar os arquivos compilados em modo “Release” em forma de publicação desejada.
   Os arquivos compilados deverão ser armazenados no servidor, em uma pasta desejada. Neste exemplo utilizaremos o diretório **“/var/www/slounge”**

6. Instalar Nginx

    Para que a aplicação esteja acessível através de requisições externas, utilizarmos o Nginx como Proxy Reverso, recebendo requisições em uma porta e as redirecionando para a porta onde a aplicação está executando localmente no servidor.

    Para instalação e configuração, execute os comandos abaixos:

    - sudo -s
    - nginx=stable # use nginx=development for latest development version
    - add-apt-repository ppa:nginx/$nginx
    - apt-get update
    - apt-get install nginx

    Em seguida devemos iniciar o servidor com o comando:

    - sudo service nginx start

Para confirmar que a o servidor está funcional e rodando, vá até um navegador e digite o endereço abaixo:

- http://<IP_do_servidor>/index.nginx-debian.html

Caso apareça a página de boas-vindas do Nginx, tudo está configurado corretamente.

7. Configurar Nginx para rotar requisições para aplicação dotnet (Proxy Reverso)

   Com o Nginx rodando, agora devemos configurar o servidor para rotear as requisições.

   - Abrir em modo edição o arquivo **/etc/nginx/sites-available/default**
   - Alterar o trecho “Server” de acordo com o código abaixo:

```
server {
listen 80;
location / {
        proxy_pass http://localhost:<porta_aplicação>;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection keep-alive;
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Forwarded-For
        $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        }
}
```

Após ajustar as configurações acima, podemos testar se estão válidas com o comando abaixo:

- sudo nginx -t

Se tudo estiver correto, devemos então recarregar o Nginx:

- sudo nginx -s reload

8. Executando e testando a aplicação

   Após realizar as configurações das etapas anteriores, devemos executar nossa aplicação dotnet para verificar se a aplicação está rodando de forma adequada e se consegue conectar com o banco de dados.

   - Navegar até a pasta onde o compilado da aplicação está armazenado
   - Executar o seguinte comando: **dotnet SLounge.Services.WebApi.dll**

A aplicação deverá iniciar corretamente e estará sendo executada na porta escolha, de acordo com o passo 3. Caso o Nginx esteja roteando as requisições corretamente, ele irá agora receber e encaminhar as solicitações para a aplicação que acabamos de executar.

Para testar, podemos realizar as seguintes chamadas em nosso navegador:

- www.<**endereco_IP_servidor**>.com/api/v1/values: Deverá voltar “Running” caso todas configurações estejam corretas.
- www.<**endereco_IP_servidor**>.com/api/v1/values/DbTest: Deverá voltar “DB running” caso todas configurações e conexão com banco estejam corretas.

9. Configurar aplicação dotnet para executar como Serviço
   Caso todas etapas tenham sido configuradas corretamente, agora devemos configurar a aplicação dotnet como um serviço, para que seja inicializada sempre junto com o boot do sistema. Para isso devemos executar os comandos abaixo:
   - Devemos criar um arquivo para gerenciar o serviço da nossa aplicação
     - sudo vi /etc/systemd/system/<**nome_desejado_servico**>.service

Ao abrir o editor, devemos utilizar o template abaixo alterando as configurações destacada em negrito:

[Unit]
Description=<**descrição da aplicação. Ex: “Minha aplicação”>**

[Service]
WorkingDirectory=<**localização do diretório dos compilados. Ex:** /home/ec2-user/CoreRESTServer**>**
ExecStart=/usr/bin/dotnet <**Localização da dll de inicio para execução de aplicação donet. Ex:** /home/ec2-user/CoreRESTServer/CoreRESTServer.dll>
Restart=always
RestartSec=10 # Restart service after 10 seconds if dotnet service crashes
SyslogIdentifier=offershare-web-app
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target

Após salvar e sair do modo de edição, devemos executar os comandos abaixo

- sudo systemctl enable <**nome_definido_serviço**>.service
- sudo systemctl start <**nome_definido_serviço>**.service
- sudo systemctl status <**nome_definido_serviço>**.service

10. Configurando Aplicação de Sincronização CSV

A aplicação de sincronização CSV é gerada a partir do compilado do projeto **SLounge.CSVBroker.HangfireDashboard.** Os passos para executar o projeto são idênticos ao anterior, portanto:

- Definir, na aplicação, a pasta onde o sistema irá consumir os arquivos despejados pelo integrador.
  - O path para consumo é encontrado no arquivo **FilesService.cs**
  - Linha 24: Path onde os arquivos serão armazenados
  - Linha 25: Path de onde os arquivos serão obtidos
- Criar uma pasta para armazenar o compilado, no servidor
- Compilar a aplicação, com o **appsettings.json** apontando para o banco corretamente
- Configurar o nginx para atuar como proxy reverso da aplicação (Deverá ser criado uma nova sessão para uma nova porta, de acordo com exemplo abaixo)

```
server {
listen 8080;
location / {
proxy_pass http://localhost:<porta_aplicação>;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection keep-alive;
proxy_set_header Host $host;
proxy_cache_bypass $http_upgrade;
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
}
}
```

- Deverá configurar a aplicação para ser executada como serviço:
  - **Voltar ao passo 9 e criar novos arquivos e configurações**

Por fim, a aplicação estará acessível através da porta determinada, lembrando que a mesma deve ser liberada pelo Firewall.