Руководство администратора программного продукта

ЕМД NET

2025 г.

Содержание

Установка ЕМД NET на ОС Linux

Скачивание дистрибутива

Дистрибутивы доступны на официальном сайте по адресу https://емдтех.рф/distr.
Выберите необходимую версию дистрибутива и нажмите на кнопку для скачивания.
Также скачать нужный дистрибутив можно по прямой ссылке:
https://емдтех.рф/distr/file/{имя файла}

Дистрибутивы ЕМД NET Runtime

Все версии ЕМД NET Runtime именуются следующим образом:
emdnet-runtime-X.X.X.zip , где X.X.X - версия.
Например, версию 8.0.12 можно скачать по прямой ссылке:
https://емдтех.рф/distr/file/emdnet-runtime-8.0.12.zip

Дистрибутивы ЕМД ASPNET

Все версии ЕМД ASPNET именуются следующим образом:
aspnetcore-emdnet-runtime-X.X.X-OS-ARC.tar.gz , где X.X.X – версия, OS – тип операционной системы, ARC – архитектура процессора.
Например, версию 8.0.13 для linux можно скачать по прямой ссылке:
https://емдтех.рф/distr/file/aspnetcore-emdnet-runtime-8.0.13-linux-x64.tar.gz

Подготовка к установке

Для скачивания и установки могут понадобиться установка дополнительных компонент.
Устанавливаем их следующими командами:

```
sudo apt install -y wget unzip
```

Переносим дистрибутив в любой каталог, либо скачиваем напрямую.
Пример скачивания ЕМД NET Runtime версии 8.0.12

```
wget –no-check-certificate https://емдтех.рф/distr/file/emdnet-runtime-8.0.12.zip
```

Пример скачивания ЕМД NET Runtime версии 8.0.12

```
wget –no-check-certificate
https://емдтех.рф/distr/file/aspnetcore-emdnet-runtime-8.0.13-linux-x64.tar.gz
```

Установка

ЕМД NET Runtime

Разархивируем архив

```
unzip emdnet-runtime-8.0.12.zip
```

Установка из дистрибутива осуществляется следующей командой:

```
dpkg -i *.deb
```

Проверить установку можно командой, которая покажет установленные версии

```
dotnet --list-runtimes
```

ЕМД ASPNET

Следующие команды нужны для задания переменной DOTNET_ROOT среды текущего рабочего каталога, за которым следует .dotnet. Этот каталог создается, если он не существует. Переменная DOTNET_FILE среды — это имя файла двоичного выпуска .NET, который требуется установить. Этот файл извлекается в DOTNET_ROOT каталог. В DOTNET_ROOT переменную среды добавляются PATH каталог и его tools подкаталог

```
DOTNET_FILE= aspnetcore-emdnet-runtime-8.0.13-linux-x64.tar.gz
export DOTNET_ROOT=$(pwd)/.dotnet
mkdir -p "$DOTNET_ROOT" && tar zxf "$DOTNET_FILE" -C "$DOTNET_ROOT"
export PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
```

Применение лицензии

Для применения лицензии нужно создать файл с лицензионным ключом в каталоге **/emd**. Например, для лицензионного ключа:
iKMC3VHgQIIL4CHHZuphu04mbmXUtxV0a6JVtLG+e2Tas6KZMaGi3Yn
gEK34YkUsnTbkYqmVG7pqKNm7iTEOn7OQSXNTXJP27NFf9PvSErtjZpuuLzK3Kox5XPn6
mIjordrmKymZU8v0AIPMVMCMHkILy5r:

```
mkdir -p /emd echo
"iKMC3VHgQIIL4CHHZuphu04mbmXUtxV0a6JVtLG+e2Tas6KZMaGi3Yn
gEK34YkUsnTbkYqmVG7pqKNm7iTEOn7OQSXNTXJP27NFf9PvSErtjZpuuLzK3K
ox5XPn6mI
```

После применения лицензии можно проверить запуск приложения, например

```
dotnet MyApp.dll
```

На экране будет выведена информация о лицензии. В случае, если лицензия просрочена, будет отображаться следующая информация:

```
EMD license id: 123
EMD license name: тестовая лицензия
EMD license Expiration: 20250120
EMD license expired
```

Работа с командной строкой

Emd NET поддерживает весь функционал dotnet.
Основные команды, которые могут потребоваться для работы:

Чтобы получить сведения о среде и доступных командах, выполните следующие действия:

```
dotnet [--version] [--info] [--list-runtimes] [--list-sdks]

dotnet -h|--help
```

Запуск приложения:

```
dotnet [--additionalprobingpath <Путь>] [--additional-deps <Путь>]
   [--fx-version <Версия>]  [--roll-forward <Настройки>]
   <Путь_до_приложения> [arguments]

dotnet exec [--additionalprobingpath] [--additional-deps <Путь>]
   [--depsfile <Путь>]
   [--fx-version <Версия>]  [--roll-forward <Настройки>]
   [--runtimeconfig <Путь>]
   <Путь_до_приложения> [arguments]
```

Команда dotnet выполняет две функции:

1) Предоставляет команды для работы с проектами .NET.
   Например, команда dotnet build выполняет построение проекта. Каждая команда определяет свои параметры и аргументы. Все команды поддерживают параметр --help, позволяющий вывести краткую справку по их использованию.

2) Запускает приложения .NET.
   Для запуска приложения необходимо указать путь к его файлу .dll. Чтобы запустить приложение, необходимо найти и выполнить точку входа, которая, в случае использования консольных приложений, является методом Main.
   Например, команда dotnet myapp.dll запускает приложение myapp.
   Дополнительные сведения о параметрах развертывания см. в статье Развертывание приложений .NET.

Параметры отображения сведений о среде и доступных команд

Следующие параметры доступны, если dotnet используется сам по себе, без указания команды или приложения для запуска. Например, *dotnet --info* или *dotnet --version*. Выводит сведения о среде.

--info

Выводит подробные сведения об установке .NET и среде компьютера, например текущую операционную систему и фиксацию SHA версии .NET.

--version

Выводит версию пакета SDK для .NET, используемого командами dotnet , на которую может повлиять файл global.json . Доступно только при установке пакета SDK.

--list-runtimes

Выводит список установленных сред выполнения .NET. Версия x86 пакета SDK содержит только среды выполнения x86, а в версии x64 пакета SDK содержатся только среды выполнения x64.

--list-sdks

Выводит список установленных пакетов SDK для .NET.

-?|-h|--help

Выводит список доступных команд.

Параметры выполнения команды

Для dotnet с командой доступны следующие параметры:
Например, *dotnet build --help* или *dotnet build --verbosity diagnostic.*

-d|--diagnostics

Включает вывод диагностических данных.

-v|--verbosity <Уровень>

Задает уровень детализации команды. Допустимые значения: q[uiet], m[inimal], n[ormal], d[etailed] и diag[nostic]. Поддерживается не во всех командах. Дополнительные сведения см. на странице соответствующей команды.

-?|-h|--help

Выводит документацию по заданной команде. Например, *dotnet build --help* отображает справку по команде build .

command options

Для каждой команды определяются относящиеся к ней параметры. Список доступных для команды параметров можно просмотреть на соответствующей ей странице.

Параметры запуска приложения

При запуске приложения в dotnet доступны следующие параметры:
Например, dotnet --roll-forward Major myapp.dll.

--additionalprobingpath <Путь>

Путь, содержащий политику проверки и проверяемые сборки. Повторите этот параметр, чтобы указать несколько путей.

--additional-deps <Путь>

Путь к дополнительному файлу .deps.json. Файл deps.json содержит список зависимостей, зависимости компиляции и сведения о версии, используемые для устранения конфликтов сборок. Дополнительные сведения см. в разделе Файлы конфигурации среды выполнения на GitHub.

--roll-forward <Настройки>

Управляет применением наката к приложению. SETTING может иметь одно из следующих значений. Если тип не указан, по умолчанию используется вариант Minor.

- *LatestPatch* — накат до версии с наибольшим номером исправления. Отключает накат дополнительных версий.
- *Minor* — накат до дополнительной версии со следующим по порядку возрастания номером, если запрошенная дополнительная версия отсутствует. Если запрошенная дополнительная версия присутствует, используется политика *LatestPatch*.
- *Major* — накат до основной версии со следующим по порядку возрастания или дополнительной версии с наименьшим номером, если запрошенная дополнительная версия отсутствует. Если запрошенная основная версия присутствует, используется политика *Minor*.
- *LatestMinor* — накат до дополнительной версии с наибольшим номером, даже если запрошенная дополнительная версия присутствует. Предназначен для сценариев размещения компонентов.
- *LatestMajor* — накат до основной версии с наибольшим номером и дополнительной версии с наибольшим номером, даже если запрошенная

основная версия присутствует. Предназначен для сценариев размещения компонентов.
- *Disable* — накат не выполняется. Привязка только к указанной версии. Эта политика не рекомендуется для общего использования, поскольку отключает возможность наката до последних исправлений. Это значение рекомендуется использовать только для тестирования.

Все параметры, кроме параметра *Disable*, будут использовать версию с последним доступным исправлением.

Поведение наката также можно настроить в свойствах файла проекта, файла конфигурации среды выполнения и переменной среды. Дополнительные сведения см. в разделе Накат основной версии среды выполнения.

--fx-version <Версия>

Версия среды выполнения .NET, используемой для запуска приложения.

Этот параметр переопределяет версию первой ссылки на платформу в файле .runtimeconfig.json приложения. Таким образом, он работает правильно только с одной ссылкой на платформу. Если приложение содержит более одной ссылки на платформы, использование этого параметра может приводить к ошибкам.

Параметры запуска приложения с exec помощью команды

Следующие параметры доступны только при dotnet запуске приложения с помощью exec команды:
Например, dotnet exec --runtimeconfig myapp.runtimeconfig.json myapp.dll.

--depsfile <Путь>

Путь к файлу deps.json. Файл конфигурации deps.json содержит информацию о зависимостях, необходимых для выполнения приложения. Этот файл создается пакетом SDK для .NET.

--runtimeconfig <Путь>

Путь к файлу runtimeconfig.json. Файл runtimeconfig.json содержит параметры времени выполнения и обычно называется <applicationname.runtimeconfig.json>.
Дополнительные сведения см. в статье Параметры конфигурации среды выполнения .NET.

Настройка сервиса приложения

Для развертки сервиса приложения используются стандартные средства ОС. Рассмотрим пошаговую инструкцию для формирования сервиса приложения

1) Перед разверткой приложения рекомендуется создать пользователя ОС от имени которого будет работать сервис.

2) В подготовленный каталог для приложения копируем все файлы вашего приложения. Если каталога еще нет, то создаем его. Рекомендуется дать права пользователю сервиса только на этот каталог

   Создать пользователя (например **emdnetusr**), под которым нельзя будет авторизоваться в ОС (--shell /bin/false) и дать ему права на каталог (например **/opt/emdnetappcat**) можно следующими командами:

   ```
   useradd -r -d /opt/emdnetappcat -m emdnetusr --shell /bin/false
   sudo -u emdnetusr mkdir /opt/emdnetappcat
   ```

3) Также желательно знать абсолютное расположение команды dotnet. Проверить можно сделать командой

   ```
   which dotnet
   ```

4) Создаем сервис (например с именем **emdnetapp**) для запускаемого файла приложения (например **EmdNetApp.dll**), если dotnet расположен в **/usr/bin/dotnet** стандартными средствами

   ```
   cat << EOF > /etc/systemd/system/emdnetapp.service
   [Unit]
   Description=EmdNetApp
   After=syslog.target network.target

   [Service]
   PermissionsStartOnly=true
   WorkingDirectory=/opt/emdnetappcat
   ExecStartPre=/usr/bin/chown -Rf emdnetusr:emdnetusr /opt/emdnetappcat
   ExecStart=/usr/bin/dotnet /opt/emdnetappcat/EmdNetApp.dll
   Restart=always
   RestartSec=10
   KillMode=process
   KillSignal=SIGINT
   StandardOutput=syslog
   StandardError=syslog
   SyslogIdentifier=emdnetusr
   User=emdnetusr
   Group=emdnetusr
   Environment=ASPNETCORE_ENVIRONMENT=Production
   Environment=ASPNETCORE_USE_XFORWARDEDFOR=true

   [Install]
   WantedBy=multi-user.target
   ```

```
EOF

systemctl enable emdnetapp
systemctl start emdnetapp
```

Подключение SSL-сертификатов к приложению

Через файл настроек приложения appsettings.json

Добавьте в appsettings.json корневую секцию Kestrel и настройте в ней Endpoints.

Endpoints представляет собой объект Endpoint, каждое свойство которого определяет точку прослушивания. Имя каждого свойства может быть произвольным.

Каждый Endpoint содержит информацию о

- прослушиваемом Url, в том числе с возможностью указания порта
- информации о сертификате Certificate
- иные настройки

Certificate можно определять разными способами

Сертификат из локального .pfx файла

```
"Kestrel": {
   "Endpoints": {
     "Https_domain": { // любое имя точки доступа
       "Url": "https://myurl", // URL который будет прослушиваться, в том числе с возможностью указания порта
       "Certificate": {
         "Path": "/path/to/cert.pfx", // Путь до файла сертификата
         "AllowInvalid": true, // Указывает, следует ли учитывать недопустимые сертификаты, например самозаверяющие сертификаты
         "Password": "certpassword" // Пароль от сертификата
       }
     }
   }
 }
```

Сертификат из локальных .pem/.crt и .key файлов

```
"Kestrel": {
   "Endpoints": {
     "Https_domain": { // любое имя точки доступа
       "Url": "https://myurl", // URL который будет прослушиваться, в том числе с возможностью указания порта
       "Certificate": {
         "Path": "/path/to/cert.pem", // Путь до файла сертификата .pem или .crt
```

```
        "AllowInvalid": true, // Указывает, следует ли учитывать
недопустимые сертификаты, например самозаверяющие сертификаты
        "KeyPath": "/path/to/cert.key", // Путь до файла ключа .key
        "Password": "certpassword"
      }
    }
  }
}
```

Из хранилища сертификатов

```
"Kestrel": {
  "Endpoints": {
    "Https_domain": { // любое имя точки доступа
      "Url": "https://myurl", // URL который будет прослушиваться, в том
числе с возможностью указания порта
      "Certificate": {
        "AllowInvalid": true, // Указывает, следует ли учитывать
недопустимые сертификаты, например самозаверяющие сертификаты
        "Subject": "SubjectName", // имя субъекта для сертификата
        "Store": "Store", // хранилище сертификатов, из которого
выполняется загрузка сертификата
        "Location": "Location" // расположение хранилища, из которого
загружается сертификат. По умолчанию CurrentUser
      }
    }
  }
}
```