ALIBABA CLOUD

阿里云

Web应用托管服务

开发指南

文档版本：20210908

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.使用Procfile配置应用进程

通常情况下，Web+对于每种技术栈类型设置了默认的启动命令，您也可以通过配置启动命令或者使用Procfile来指定，Web+将优先使用Procfile中的启动命令。

Procfile

如果需要使用Procfile，请在源包根目录中包含一个名为Procfile的文件，格式如下：

```
web: <command>
```

设置示例：

```
web: /home/admin/app/demo/startup.sh
```

> 注意
>
> - Procfile中的每行内容都必须符合以下正则表达式：^[A-Za-z0-9_]+:\s*.+$。
> - Web+将识别开头为 `web:` （冒号后面有一个空格）的命令，并以此作为服务的启动命令。
> - 此命令必须一直在前台运行，命令执行结束时服务即停止。
> - 启动命令的工作目录为应用部署包的根目录。
> - 启动命令将以admin用户身份执行。

2.Tomcat

2.1. 项目文件夹结构设置

Tomcat服务器中使用WAR包来部署应用时，WAR包的项目结构必须符合一定的标准，一个Web项目目录中包含Web应用程序代码和配置文件以及静态文件等。

项目文件夹结构

为了简化工程的编译和打包步骤，推荐使用下面的项目文件层次结构。

```
tomcat-webapp
└─── src
  └─── main
    ├─── java          - 源代码目录
    │  └─── com
    │    └─── demoapp
    │      └─── Hello.java
    ├─── resources       - 资源配置文件
    │  └─── application.properties
    └─── webapp
      ├─── 404.jsp        - 404jsp页面
      ├─── WEB-INF        - 安全目录
      │  ├─── classes       - 已编译的类
      │  ├─── lib         - jar库文件
      │  │  └─── mysql-connector-java-8.0.8-dmr.jar
      │  ├─── views       - 页面模板
      │  │  └─── index.mustache
      │  └─── web.xml       - 部署描述文件
      ├─── index.jsp       - jsp页面
      └─── static         - 静态资源文件
        ├─── css         - css样式资源文件
        │  └─── demoapp.css
        ├─── fonts        - 字体资源文件
        ├─── images        - 图像资源文件
        │  └─── demoapp.png
        └─── js          - JavaScript文件
          └─── bootstrap.min.js
```

*src/main/java*目录下的内容包含您开发的应用程序，即未编译的Java类文件，这些类将被编译成可通过应用程序代码访问的*.class*文件并置于*src/main/webapp/WEB-INF/classes*目录中。Java类文件编译完成后，编译好的class文件将被置于*webapp/WEB-INF/classes*中，并与webapp目录一并被打包部署到服务器上。

webapp根目录文件夹结构

webapp的根目录中存储了HTML页面、JSP页面以及静态资源等内容，这些内容将与WEB-INF一起被打包部署到服务器上。

webapp中除了WEB-INF外的其他内容可通过客户端直接访问，如404.jsp页面和index.jsp页面。static目录中存放CSS样式文件，图片文件和JavaScript文件等可由客户端访问的资源。

```
webapp
├──── 404.jsp
├──── WEB-INF
├──── index.jsp
└──── static
  ├──── css
  │  └──── demoapp.css
  ├──── fonts
  ├──── images
  │  └──── demoapp.png
  └──── js
    └──── bootstrap.min.js
```

WEB-INF子目录文件夹结构

webapp目录下的WEB-INF子目录是Java的Web应用安全目录，客户端无法直接访问其中的内容，但服务端可以访问。WEB-INF子目录包含以下内容：

- classes目录：包含您开发的应用程序源码编译后的.class文件。
- lib目录：包含Web应用依赖的各种JAR文件，如数据库驱动JAR文件。
- views目录：包含页面模板文件，例如mustache文件。
- web.xml文件：Web应用配置文件。

```
WEB-INF
├──── classes      - 已编译的类
├──── lib          - JAR库文件
├──── views        - 页面模板
└──── web.xml      - Web应用配置文件
```

2.2. 使用Web+部署Tomcat应用

Web+的Tomcat技术栈是一组环境配置，用于Tomcat Web容器中运行的Java Web应用。在本教程中，您将学习如何开发一个简单的Tomcat应用，并将其部署到Web+。

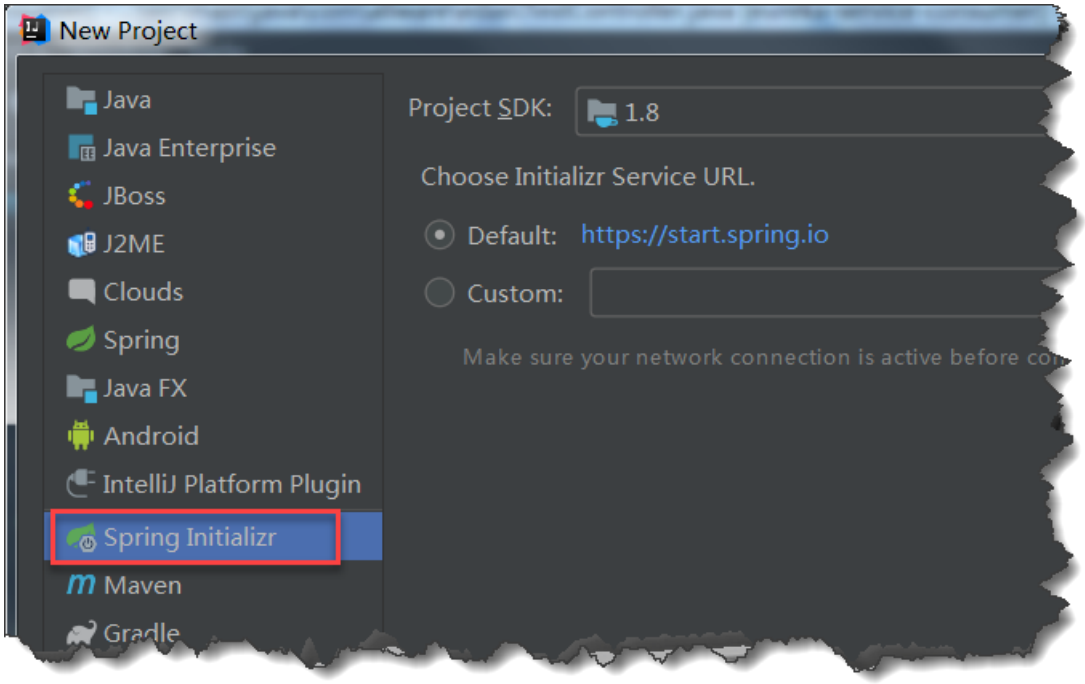
准备工作

在进入本教程之前，请确保您已经安装并配置好了以下工具和容器：

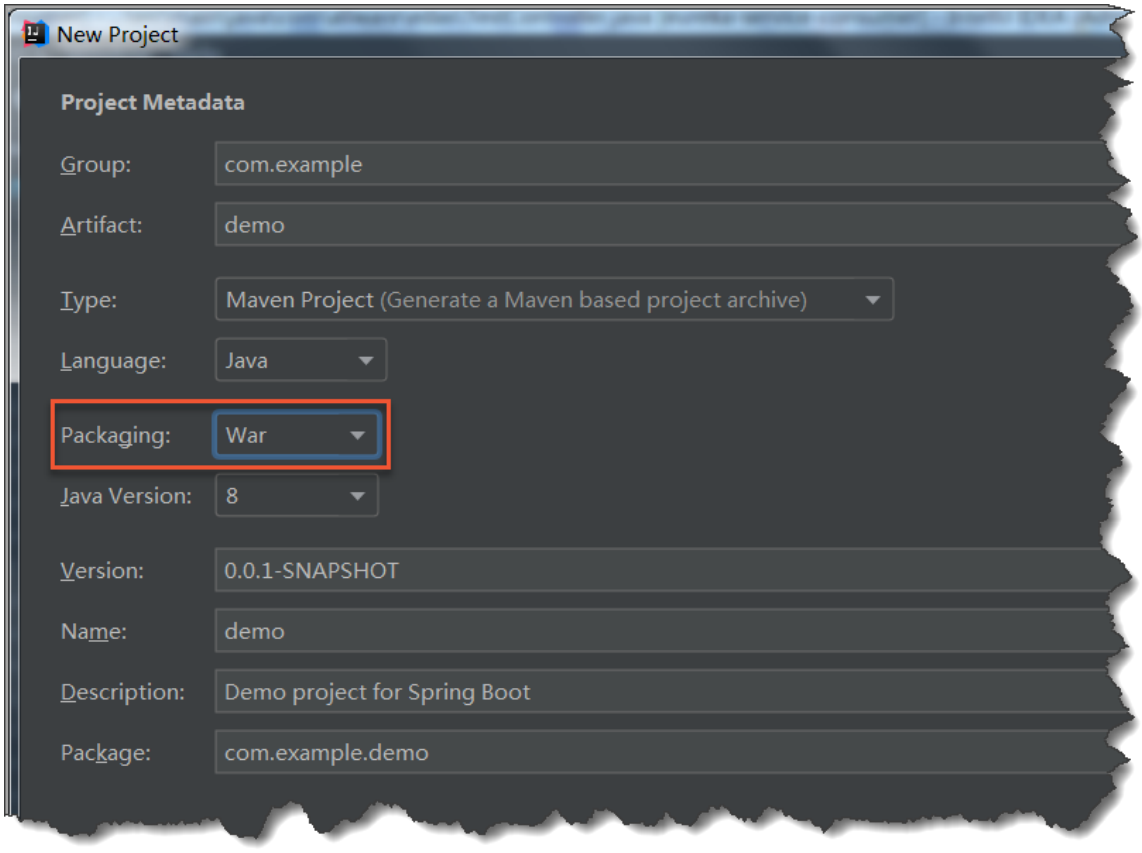
- IntelliJ IDEA
- Maven
- JDK
- Tomcat 8.5及以上版本

步骤一：创建Tomcat Demo工程

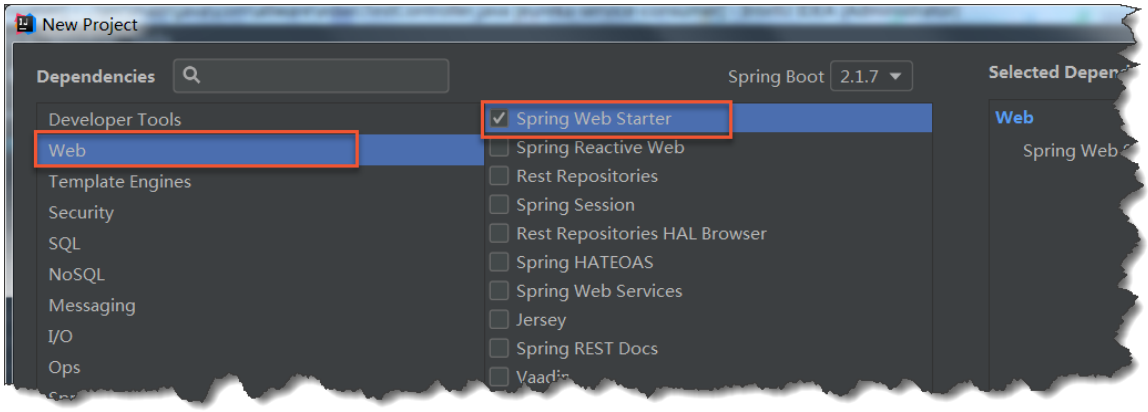
1. 启动IntelliJ IDEA。
2. 选择**File->New->Project**，新建一个工程。
3. 选择**Spring Initializr**，然后单击**Next**。



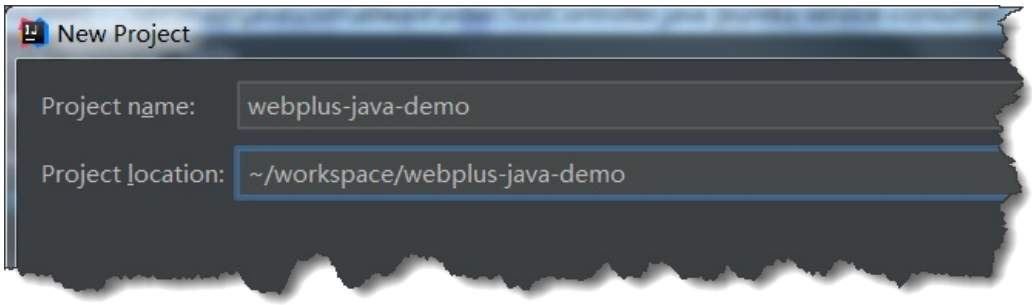
4. 输入工程信息，并选择打包方式为*War*，设置完成后单击**Next**。



5. 在**Dependencies**页面单击**Web**并勾选**Spring Web Starter**，然后单击**Next**。

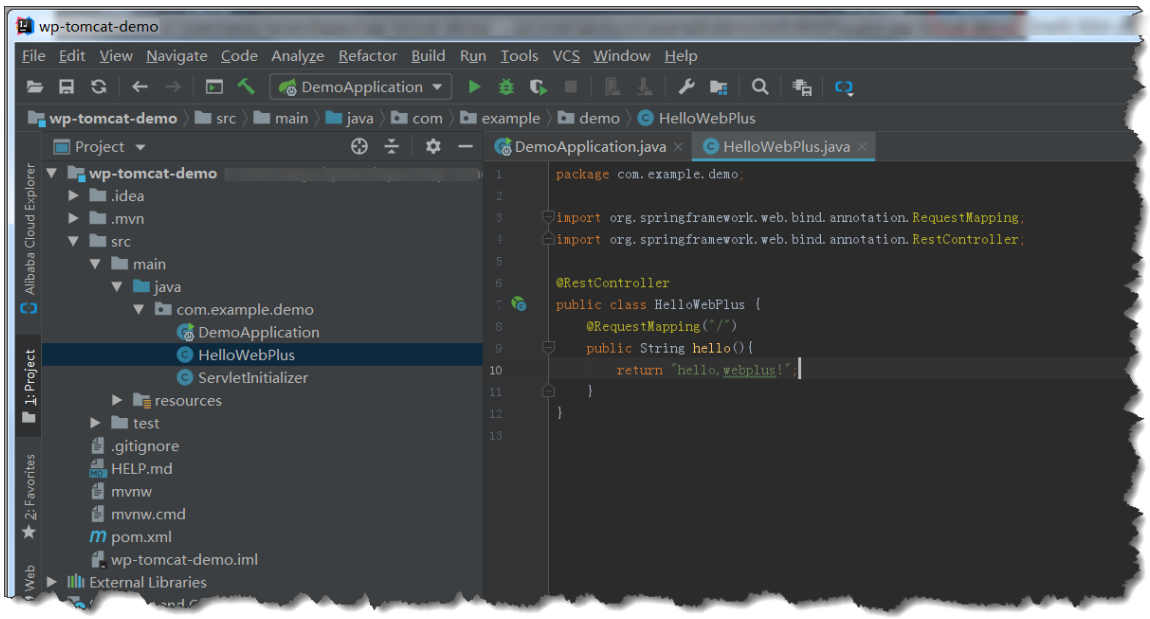


6. 输入工程名称，并单击**Finish**完成创建。



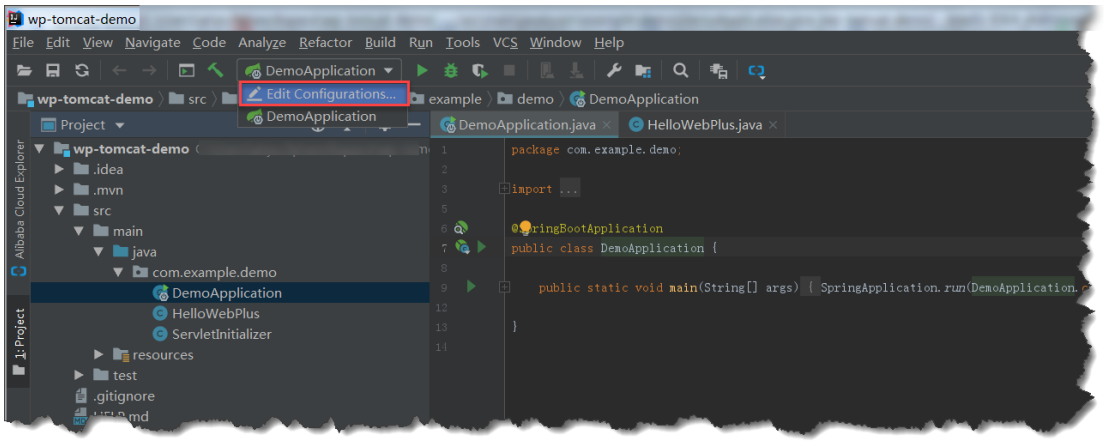
步骤二：配置应用

1. 新建一个Controller类。

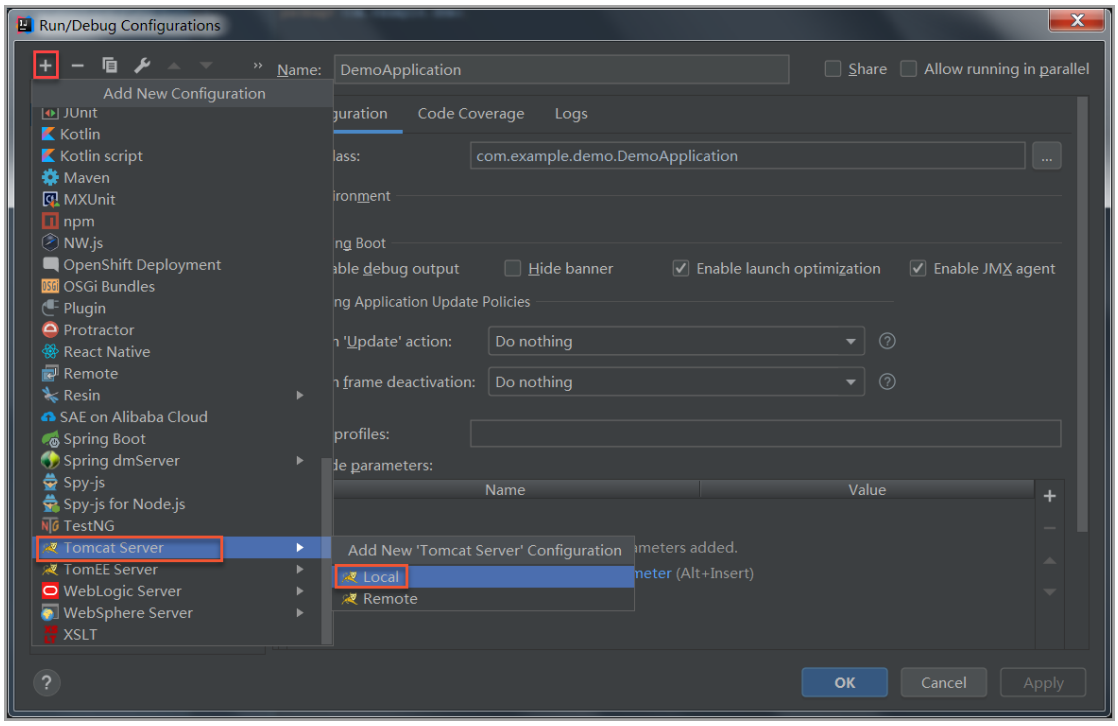


2. 参照以下步骤，配置本地运行环境。

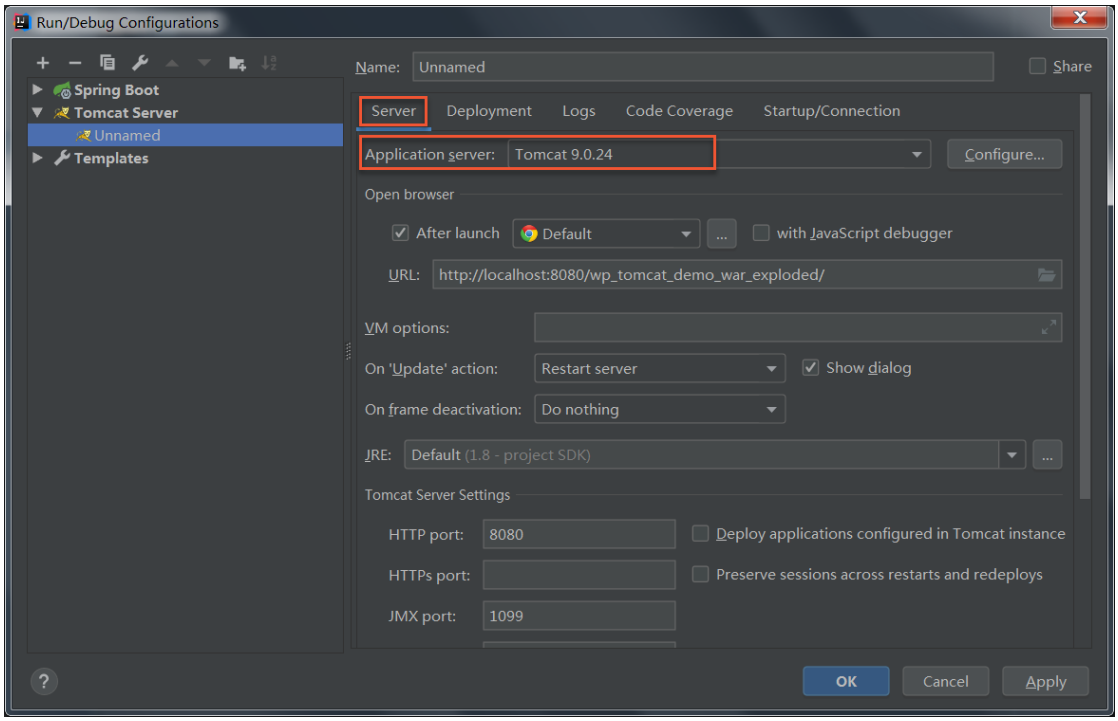
i. 在项目页面选择DemoApplication> Edit Configurations。



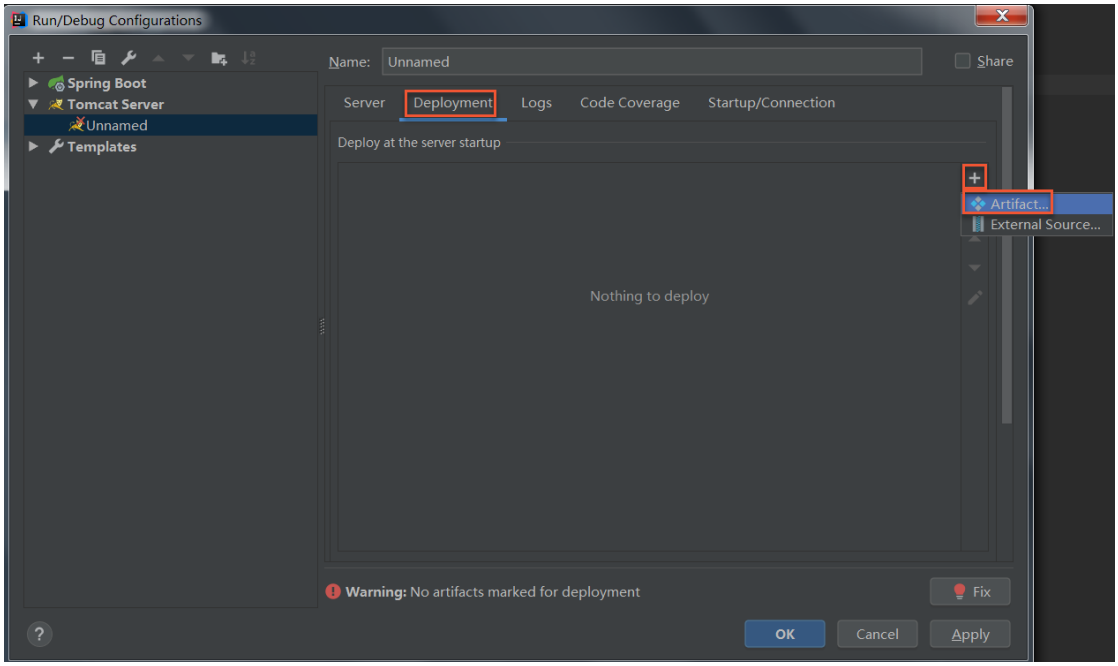
ii. 在页面左上角单击+，然后选择Tomcat Server > Local。



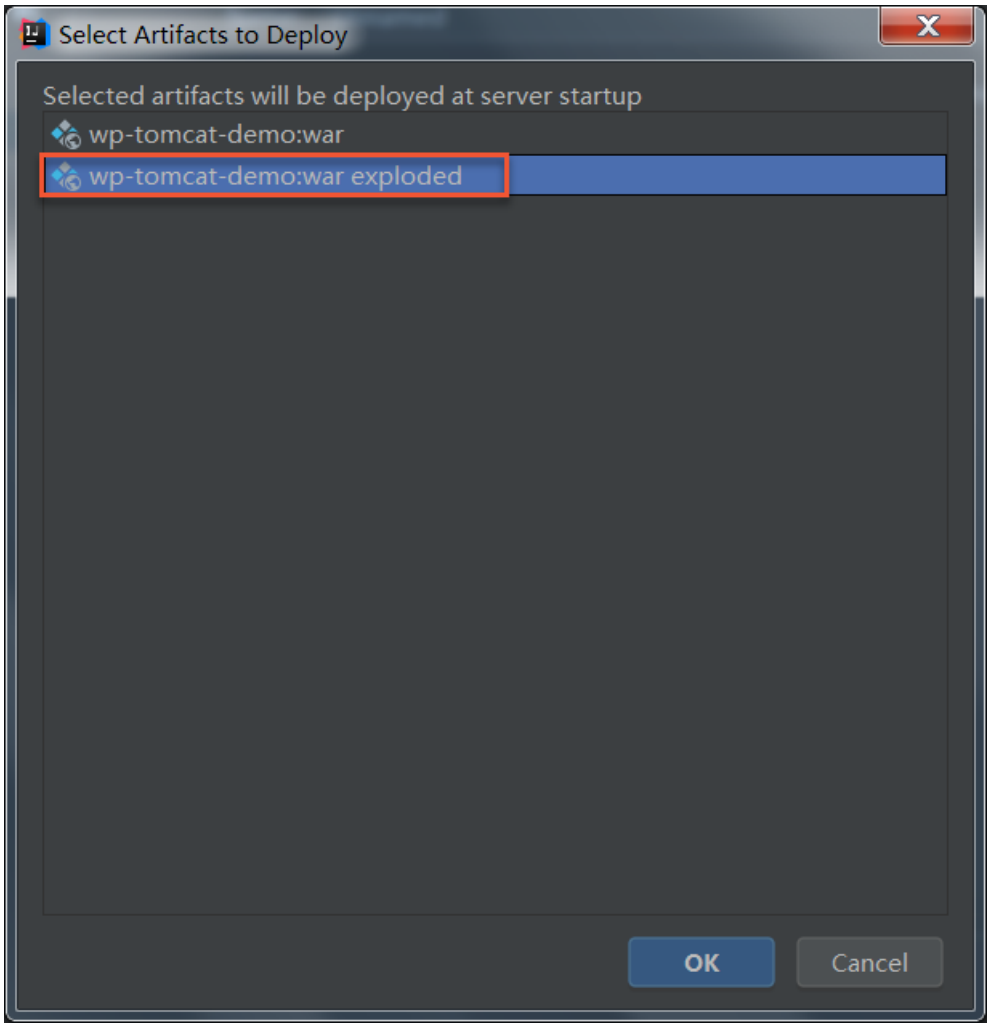
iii. 在Server页签配置8.5及以上版本的Tomcat。



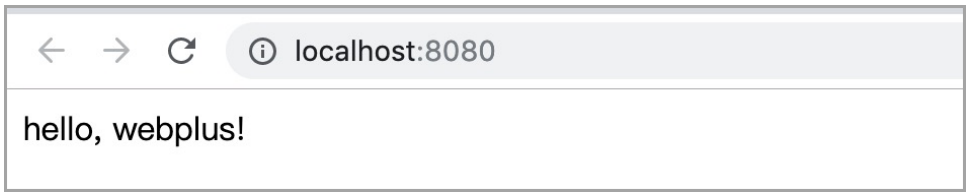
iv. 在Deployment页签选择+ > Artifacts配置部署方式。



v. 在**Select Artifacts to Deploy**对话框中选择部署方式为**war exploded**，然后单击**OK**。

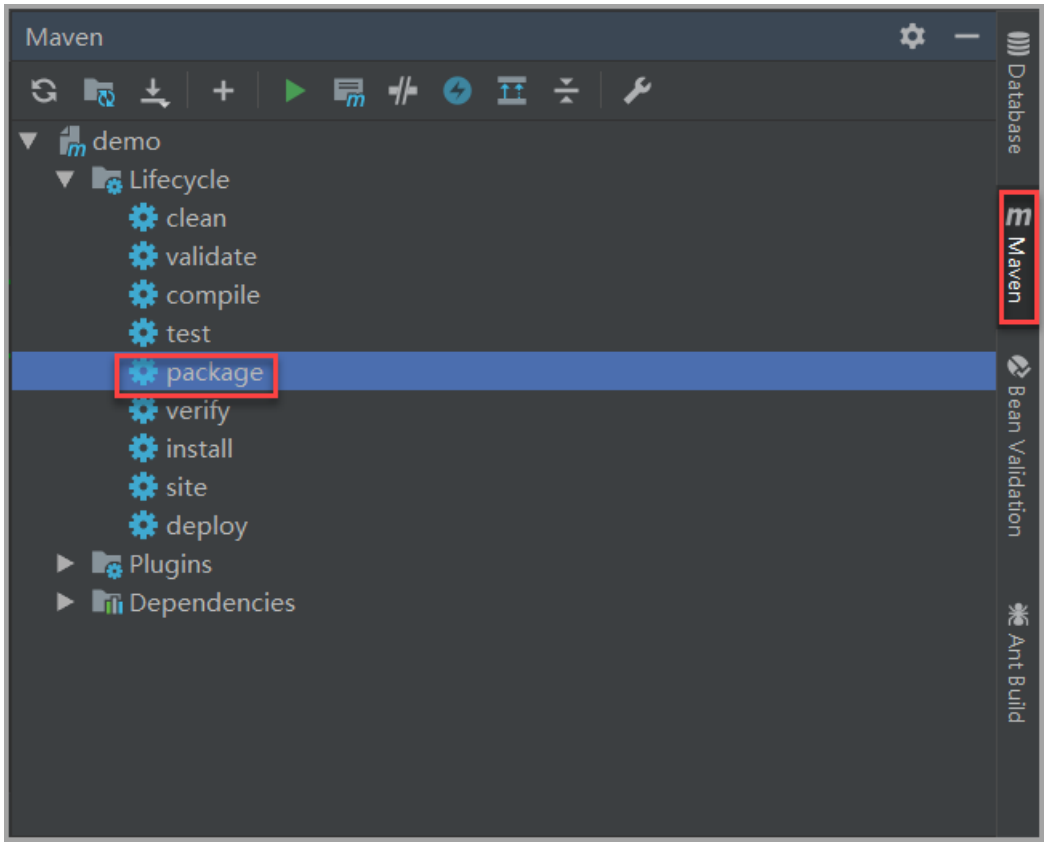


3. 单击调试、Run或Debug按钮，启动应用。
4. 在浏览器中输入*http://localhost:8080*来访问应用。

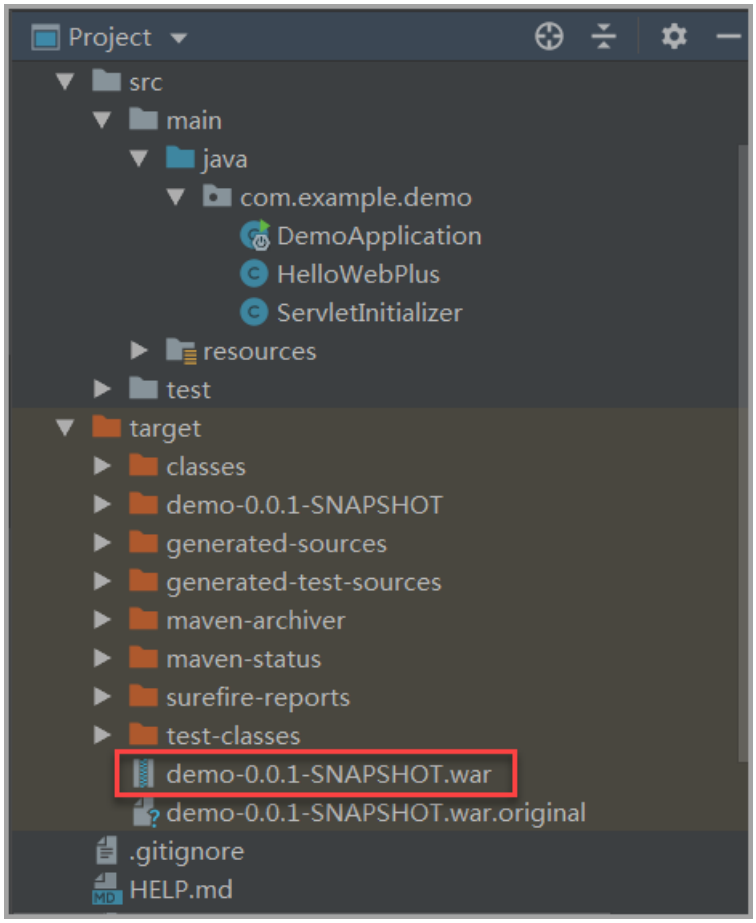


步骤三：打包应用

1. 单击右侧的**Maven**选项卡，在弹出的页面中选择**Demo > Lifecycle > Package**进行打包。



2. 打包完成后可在工程的target目录下看到一个war包（如demo-0.0.1-SNAPSHOT.war），接下来需要将这个应用程序部署到Web+应用中。



步骤四：创建应用并完成部署

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为Tomcat，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的demo-0.0.1-SNAPSHOT.war，设置部署包版本后单击**完成创建**。
5. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。

连接数据库

Tomcat类型的应用程序的连接数据库方法同Java类型的应用程序的访问数据库方式基本一致，具体操作请参见向Java应用的部署环境中添加RDS实例，连接数据库的方法示例可参考部署包alibabacloud-webplus-tomcat-demo。

更多信息

- 在Web+控制台快速部署应用的视频演示请参见在Web+控制台创建应用和部署环境。
- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见使用CLI快速部署Java应用。
- 完成应用托管之后的应用的管理操作请参见应用详情概览。
- 管理应用所在的部署环境的操作请参见管理部署环境。

3.Java

3.1. 设置Java开发环境

在本地开发环境测试Java应用，需准备好相关的开发环境。本文将介绍Java开发环境的设置步骤，并提供相关工具的安装链接。

安装Java开发工具包

选择您的开发平台，参考以下方式安装JDK。

Linux

1. 在Oracle官网下载二进制安装包，例如*jdk-8uversion-linux-x64.tar.gz*。
2. 进入JDK安装包所在目录。
3. 执行以下命令解压安装包：

   ```
   tar zxvf jdk-8uversion-linux-x64.tar.gz
   ```

4. 按以下方式配置环境变量。
   i. 执行以下命令打开配置文件。

      ```
      vim ~/.bashrc
      ```

   ii. 在配置文件中添加以下内容。

      ```
      JAVA_HOME=/Java安装路径
      CLASSPATH=$JAVA_HOME/lib/
      PATH=$PATH:$JAVA_HOME/bin
      export PATH JAVA_HOME CLASSPATH
      ```

   iii. 执行以下命令使配置生效。

      ```
      source ~/.bashrc
      ```

   iv. 检查是否安装成功。

      ```
      java -version
      javac -version
      ```

macOS

1. 在Oracle官网下载安装包，例如jdk-8uversion-macosx-x64.dmg。
2. 进入JDK安装包所在目录，双击安装包按照提示指令进行安装。
3. 按以下方式配置环境变量。
   i. 执行以下命令打开配置文件。

      ```
      vim ~/.bashrc
      ```

ii. 在配置文件中添加以下内容。

```
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_151.jdk/Contents/Home
CLASSPAHT=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
PATH=$JAVA_HOME/bin:$PATH:
export JAVA_HOME
export CLASSPATH
export PATH
```

iii. 执行以下命令使配置生效。

```
source ~/.bashrc
```

iv. 检查是否安装成功。

```
java -version
javac -version
```

Windows

1. 在Oracle官网下载安装包，例如*jdk-8version-windows-x64.exe*。
2. 进入JDK安装包所在目录，双击安装包按照提示指令进行安装。
3. 按以下方式配置环境变量。
   i. 设置JAVA_HOME环境变量。

   ```
   C:\Program Files\Java\jdk1.8.0
   ```

   ii. 修改环境变量PATH，在PATH变量后添加如下内容

   ```
   {系统中原PATH环境变量};%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin
   ```

   iii. 添加CLASSPATH环境变量。

   ```
   %JAVA_HOME%\lib;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;
   ```

   iv. 检查是否安装成功。

   ```
   java -version
   ```

安装Tomcat

访问Apache Tomcat，根据**Apache Tomcat Versions**的说明下载合适的Tomcat版本。

安装IDE

集成开发环境（IDE）是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面等工具。如果您还没使用IDE进行过Java开发，请根据个人开发习惯下载安装Eclipse或IntelliJ IDEA，下载链接如下：

- Eclipse
- IntelliJ IDEA

3.2. 使用Spring Boot开发应用

Spring Boot是一个轻量级框架，可以用来轻松地创建独立的、生产级的、基于Spring且能直接运行的应用。在本教程中，您将学习如何开发一个简单的Spring Boot应用，并将其部署到Web+。

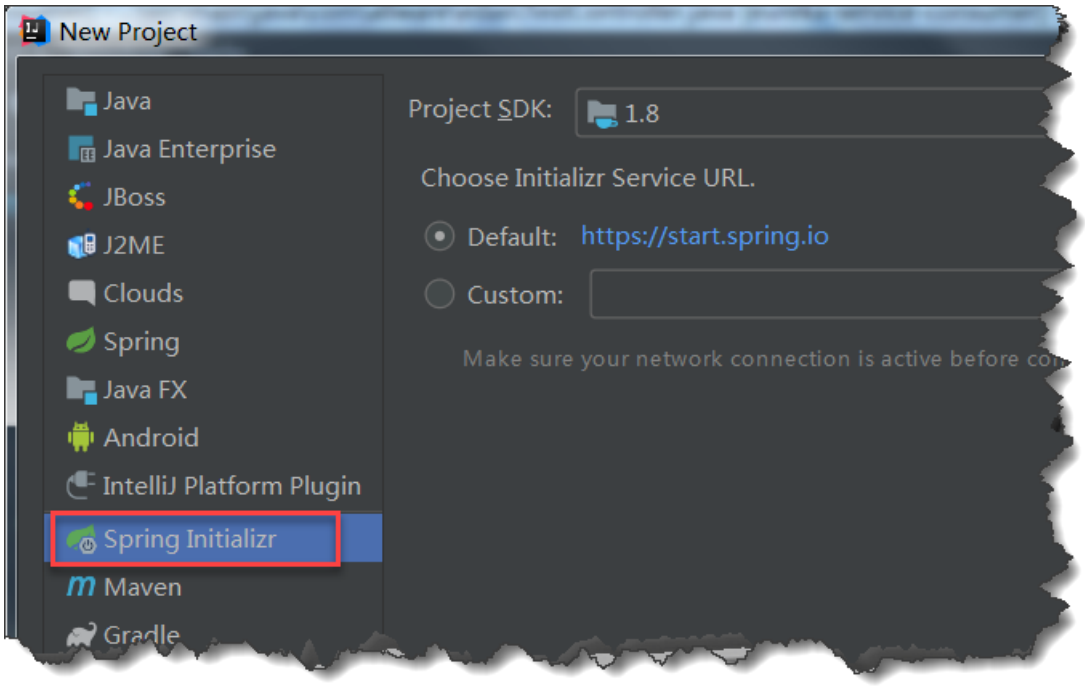
准备工作

在进入本教程之前，请确保您已经安装并配置好了以下3个工具：

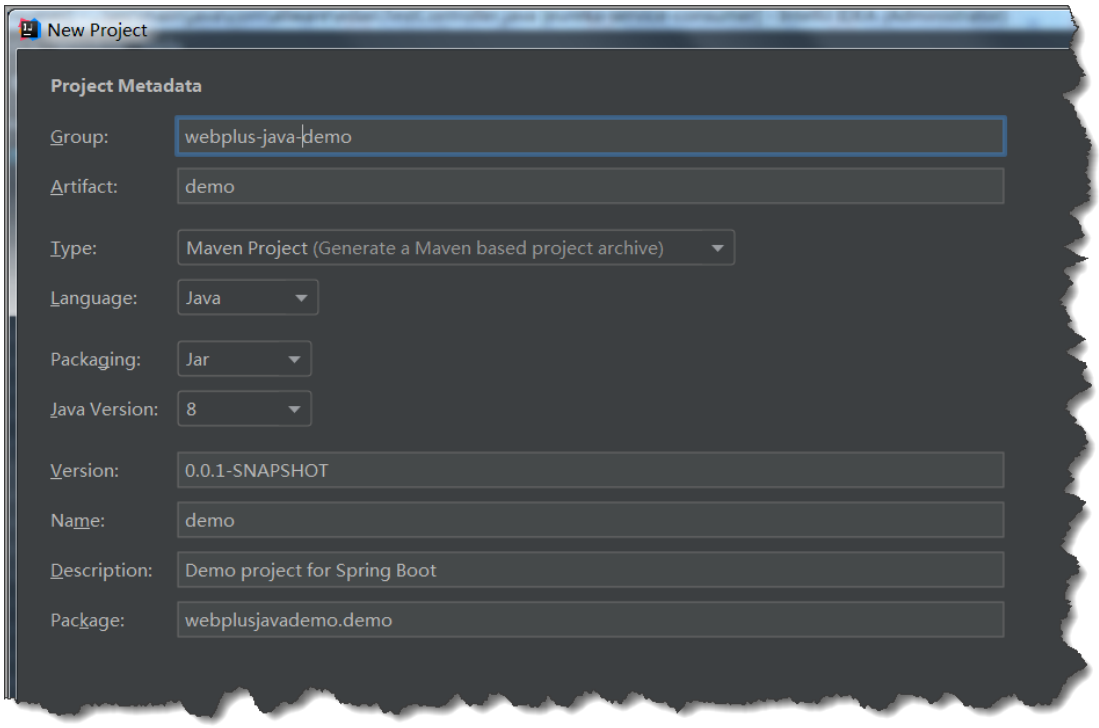
- IntelliJ IDEA
- Maven
- JDK

步骤一：创建Demo工程

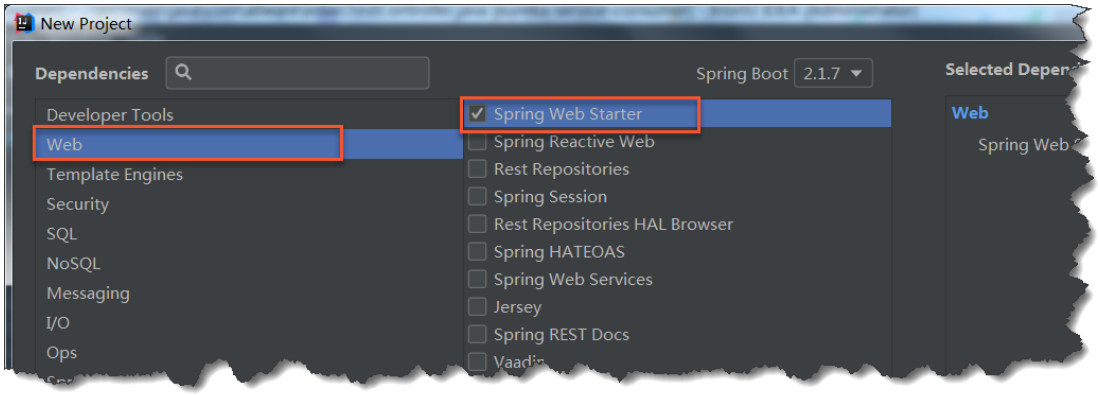
1. 启动IntelliJ IDEA。
2. 选择**File->New->Project**，新建一个工程。
3. 对项目进行配置并完成创建。
   i. 在左侧导航栏选择**Spring Initializr**，然后单击**Next**。



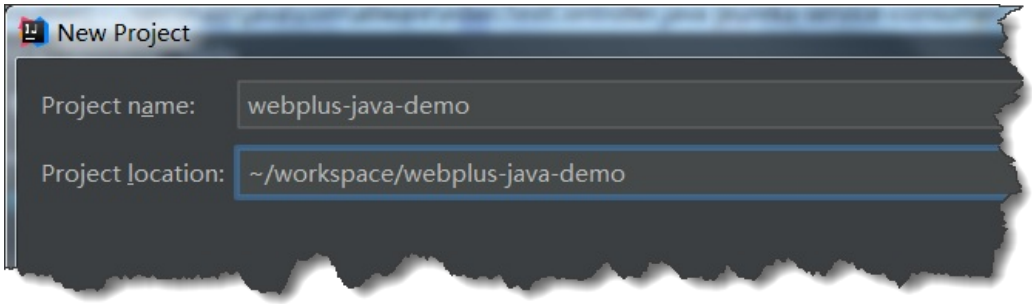
ii. 设置工程信息，设置完成后单击**Next**。



iii. 在**Dependencies**页面单击**Web**并勾选**Spring Web Starter**，然后单击**Next**。



iv. 输入工程名称，并单击**Finish**完成创建。



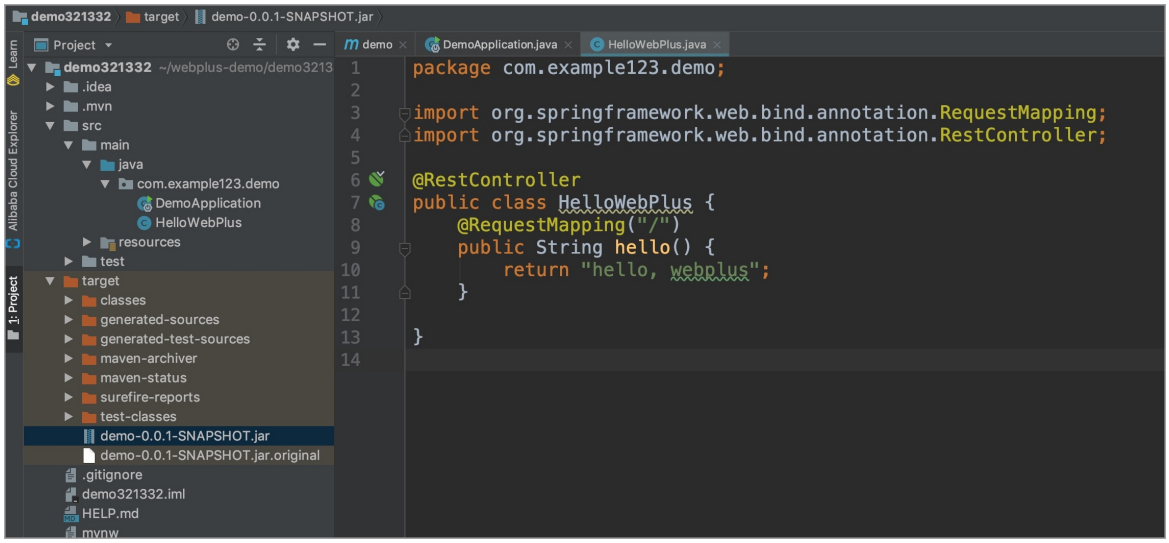
v. 打开工程目录下的pom.xml文件，并在其中加入下图所示圈注部分的配置。

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSch
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.6.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>webplus-java-demo</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>
```

步骤二：配置应用

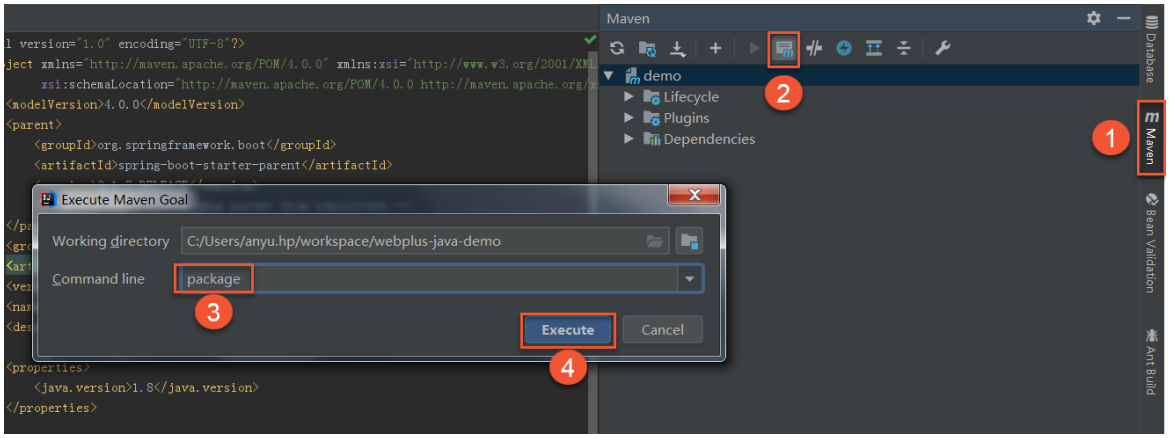
1. 新建一个Controller类。



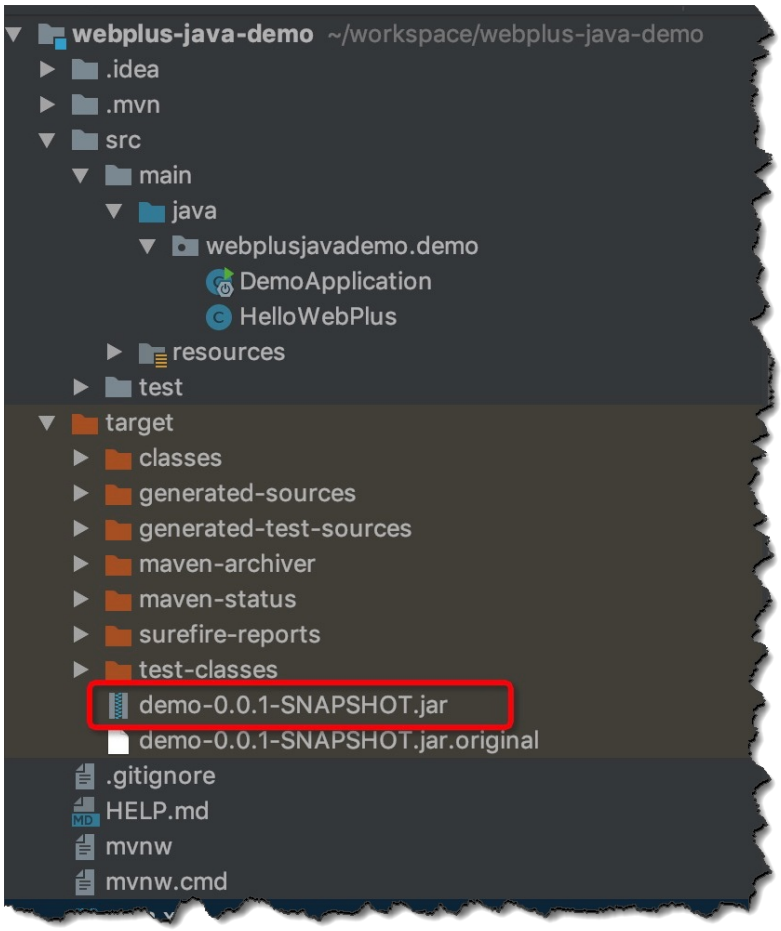
2. 单击调试、Run或Debug按钮，启动应用。
3. 在浏览器中输入*localhost:8080*来访问应用。

步骤三：构建可执行JAR

1. 单击右侧的Maven选项卡，在弹出的页面中单击Execute Maven Goal按钮。在弹出的对话框中输入package命令，然后单击Execute。



2. 打包完成后可在工程的target目录下看到一个Jar包（如demo-0.0.1-SNAPSHOT.jar），接下来需要将这个应用程序部署到Web+应用中。



步骤四：创建应用并完成部署

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为Java，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的demo-0.0.1-SNAPSHOT.jar，设置部署包版本后单击**完成创建**。

5. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。

更多信息

- 在Web+控制台快速部署应用的视频演示请参见在Web+控制台创建应用和部署环境。
- 在控制台部署应用的详细配置步骤请参见部署应用。

- 使用CLI完成应用创建和部署的操作请参见使用CLI快速部署Java应用。
- 完成应用托管之后的应用的管理操作请参见应用详情概览。
- 管理应用所在的部署环境的操作请参见管理部署环境。

3.3. 向Java应用的部署环境中添加RDS实例

您可以使用阿里云云数据库RDS的数据库实例来存储应用中需持久保存的数据。本文以给基于Spring Boot框架开发的Java应用添加RDS MySQL数据库为例，展示如何为Java应用添加数据库并验证应用与数据库是否连接。

环境变量

Web+会将数据库连接的相关信息存放在环境变量中，以便应用进行读取，相关环境变量请参考下表。

变量名	变量值	变量说明
WP_RDS_ENGINE	MySQL	RDS数据库引擎
WP_RDS_CONNECTION_ADDRESS	rm-***.mysql.rds.aliyuncs.com	RDS内网连接地址
WP_RDS_PORT	3306	RDS端口号
WP_RDS_ACCOUNT_NAME	webplus	RDS数据库账号名称
WP_RDS_ACCOUNT_PASSWORD	自定义	RDS账号密码
WP_RDS_DATABASE	webplus	RDS数据库

添加依赖和修改配置文件

1. 打开SpringBoot工程中的*pom.xml*文件，添加JDBC依赖和MySQL依赖。

```
<!-- JDBC依赖 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<!-- MySQL依赖 -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>
```

2. 打开工程中的application.properties配置文件，您可以按以下方式使用环境变量配置JDBC的连接参数，其中NONE可以修改为默认数值。

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://${WP_RDS_CONNECTION_ADDRESS:NONE}:${WP_RDS_PORT:3306}/rdsitem?useUnicode=true&characterEncoding=utf-8
spring.datasource.username=${WP_RDS_ACCOUNT_NAME:NONE}
spring.datasource.password=${WP_RDS_ACCOUNT_PASSWORD:NONE}
```

连接数据库

修改依赖和配置后，启动应用时SpringBoot会根据配置文件自动连接数据库，下面的代码片段展示了如何在Web+中的应用访问数据库。

```
  @Autowired
  private JdbcTemplate jdbcTemplate;
  // 从数据库item中查询
  public List<Item> fetchItems() {
    final String sql="select id,title,completed from item";
    RowMapper<Item> rowMapper=new BeanPropertyRowMapper<>(Item.class);
    return jdbcTemplate.query(sql, rowMapper);
  }
```

其中，Item类定义如下：

```
public class Item {
  private String id;
  private String title;
  private boolean completed;
  public String getId() {
    return id;
  }
  public void setId(String id) {
    this.id = id;
  }
  public String getTitle() {
    return title;
  }
  public void setTitle(String title) {
    this.title = title;
  }
  public boolean getCompleted() {
    return completed;
  }
  public void setCompleted(boolean completed) {
    this.completed = completed;
  }
  Item() {
  }
  @Override
  public String toString() {
    return id + " " + title + " " + completed;
  }
}
```

4.Node.js

4.1. 设置Node.js开发环境

在本地开发环境测试Node.js应用，需要准备相关的开发环境。本文将介绍Node.js开发环境的设置步骤，并提供相关工具的安装页面链接。

安装Node.js

在Node.js官方网站下载安装包。

> **说明** 为了和Web+的技术栈版本更好兼容，建议您下载Node.js 10.16.x或Node.js 8.16.x。

Linux

1. 进入Node.js安装包（例如node-v10.16.3-linux-x64.tar.xz）所在目录，执行以下命令将安装包解压到*/usr/local*目录下。

   ```
   sudo tar -C /usr/local -xzf node-v10.16.3-linux-x64.tar.xz
   ```

2. 执行以下命令创建软链接*/usr/local/node*指向刚解压的安装包路径。

   ```
   sudo ln -s /usr/local/node-v10.16.3-linux-x64 /usr/local/node
   ```

3. 解压后将可执行文件目录配置到*Path*环境变量，将以下命令添加到*$HOME/.profile*。

   ```
   export PATH=$PATH:/usr/local/node/bin
   ```

4. 执行以下命令使环境变量立即生效。

   ```
   source $HOME/.profile
   ```

5. 使用以下命令验证Node.js是否安装成功。

   ```
   node --version && npm --version
   ```

   如果显示如下信息，则说明安装包已成功安装。

   ```
   v10.16.0
   6.9.0
   ```

macOS

执行以下命令使用*brew*来快速安装Node.js。

```
brew update && brew install node
```

Windows

进入Node.js安装包所在目录，运行下载的.msi文件即可安装，无需其他配置。

安装IDE

集成开发环境（IDE，Integrated Development Environment ）是用于提供应用开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面等工具，可以显著提高开发效率。以下是Node.js开发中常用的IDE，但下列IDE可能需要通过安装插件来支持Node.js开发。

- Visual Studio Code
- Atom
- WebStorm（商业软件）

4.2. 将Express应用部署到Web+

Express是一个快速开发Node.js应用的Web框架，可以用来快速开发API、Web、后端服务等各种应用。本文档介绍了如何开发一个简单的Express应用，并将其部署至Web+。

前提条件

已配置好了Node.js的开发环境，相关操作请参见设置Node.js开发环境。

步骤一：安装express-generator

本文将使用express-generator来快速生成Express项目。请执行以下命令安装express-generator。

```
npm install -g express-generator
```

> **说明** 如果您安装了Node.js 8.2.0及以上版本，可跳过此步在创建应用时直接使用npx命令运行express-generator。

步骤二：创建应用

执行以下命令创建名为*webplus-express-app*的应用。

```
express webplus-express-app
```

> **说明** 如果您安装了Node.js 8.2.0及以上版本，可执行 `npx express-generator webplus-express-app` 命令直接运行express-generator而无需安装。

执行上述命令会创建一个名为*webplus-express-app*的目录，结构如下：

```
webplus-express-app/
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade
```

步骤三：安装本地依赖

1. 执行以下命令进入创建好的应用目录。

   ```
   cd webplus-express-app
   ```

2. 执行以下命令安装本地依赖。

   ```
   npm install
   ```

步骤四：在本地运行应用

1. 执行以下命令在本地运行应用，以验证其是否可以正常工作。

   ```
   npm start
   ```

   当您看到命令行页面上显示以下信息时，则表示应用启动成功。

   ```
   > webplus-express-app@0.0.0 start /home/admin/webplus-express-app
   > node ./bin/www
   ```

2. 查看运行结果：
   - 在浏览器中输入http://localhost:3000来访问应用。

   - 执行curl http://localhost:3000命令，查看返回的运行结果：

```
<!DOCTYPE html>
<html>
  <head>
    <title>Express</title>
    <link rel="stylesheet" href="/stylesheets/style.css">
  </head>
  <body>
    <h1>Express</h1>
    <p>Welcome to Express</p>
  </body>
</html>
```

说明 为方便查看，此处对返回的信息进行了格式化，原始结果是一整行文本。

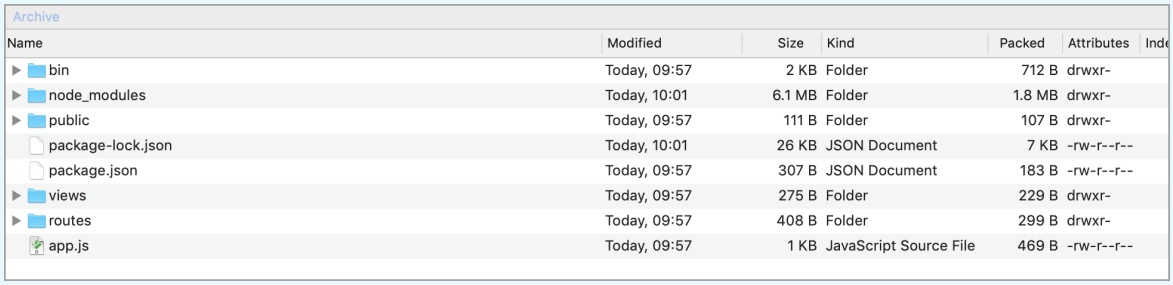
3. 查看应用的运行结果之后，可以使用*CTRL+C*停止服务。

步骤五：打包应用

执行以下命令将上面生成的应用的项目工程打包。

```
zip -r webplus-express-app.zip .
```

注意 生成的部署包需包含*node_modules*目录，且压缩包不能包含第一级目录，打包示例如下：



步骤六：将应用部署至Web+

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为Node.js，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的webplus-express-app.zip，设置部署包版本后单击**完成创建**。
5. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。

4.3. 向Node.js应用部署环境中添加RDS数据库实例

您可以添加RDS数据库实例来存储应用中需持久保存的数据。本文在部署了node.js英语的基础上，展示如何为Node.js应用添加数据库并读写其中的数据。

环境变量

Web+会将数据库连接的相关信息存放在环境变量中，以便应用进行读取，相关环境变量请参考下表。

变量名	变量值	变量说明
WP_RDS_ENGINE	MySQL	RDS数据库引擎
WP_RDS_CONNECTION_ADDRESS	rm-***.mysql.rds.aliyuncs.com	RDS内网连接地址
WP_RDS_PORT	3306	RDS端口号
WP_RDS_ACCOUNT_NAME	webplus	RDS数据库账号名称

变量名	变量值	变量说明
WP_RDS_ACCOUNT_PASSWORD	自定义	RDS账号密码
WP_RDS_DATABASE	webplus	RDS数据库

添加数据库驱动

进入应用的项目工程目录，例如进入将Express应用部署到Web+应用的*webplus-express-app*目录，执行以下命令添加MySQL数据库驱动。

```
npm install mysql
```

添加数据库访问功能

打开*routes/users.js*文件，修改如下：

```
var express = require('express');
var mysql = require('mysql');
var router = express.Router();
router.get('/', function(req, res, next) {
 var connection = mysql.createConnection({
  host: process.env.WP_RDS_CONNECTION_ADDRESS,
  user: process.env.WP_RDS_ACCOUNT_NAME,
  password: process.env.WP_RDS_ACCOUNT_PASSWORD,
  database: process.env.WP_RDS_DATABASE
 });
 connection.connect();
 connection.query('SELECT "Tom" AS user_name', function(error, results) {
  if (error) {
   throw error;
  }
  res.send('User name queried from database: ' + results[0].user_name);
 });
});
module.exports = router;
```

更多信息

关于如何使用Web+来管理RDS，可参考云数据库RDS。

5.Go

5.1. 设置Go开发环境

在本地开发环境测试Go应用，需准备好相关的开发环境。本文将介绍Go开发环境的设置步骤，并提供相关工具的安装页面链接。

安装Go

根据您的开发平台，从Go官方网站下载对应的版本，对于各个主流平台，Go都有很完善的支持，下载完成后请分别参考以下方式安装。

Linux

1. 使用以下命令将安装包解压到*/usr/local/go*目录下：

   ```
   tar -C /usr/local -xzf go$VERSION.$OS-$ARCH.tar.gz
   ```

2. 解压后将可执行文件目录配置到PATH环境变量，将以下命令添加到*/etc/profile*或*$HOME/.profile*。

   ```
   export PATH=$PATH:/usr/local/go/bin
   ```

3. 执行以下命令使环境变量立即生效。

   ```
   source $HOME/.profile
   ```

macOS

执行以下命令使用brew来快速安装Go。

```
brew update && brew install go
```

Windows

进入Go安装包所在目录，运行下载的.msi文件即可安装，无需其他配置。

配置GOPATH

环境变量*GOPATH*标识了工作区的目录，通常情况下您需要设定此环境变量来指定工作区目录。各个平台对应设置方法请参见文档Setting GOPATH。

安装IDE

集成开发环境（IDE，Integrated Development Environment ）是用于提供应用开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面等工具，可以显著提高开发效率。以下是Go开发中常用的IDE，但下列IDE可能需要通过安装插件来支持Go开发。

- Eclipse
- Visual Studio Code
- GoLand（商业软件）

5.2. 将Beego应用部署到Web+

Beego是一个快速开发Go应用的HTTP框架，可以用来快速开发API、Web、后端服务等各种应用。本文档介绍了如何开发一个简单的Beego应用，并将其部署至Web+。

步骤一：安装Beego

1. 执行以下命令来安装Beego和工具bee。

```
go get -u github.com/astaxie/beego
go get -u github.com/beego/bee
```

2. 执行以下命令将*$GOPATH/bin*目录加入$PATH环境变量。

```
echo 'export PATH="$GOPATH/bin:$PATH"' >> ~/.profile
source >> ~/.profile
```

步骤二：创建应用

1. 打开终端，进入*$GOPATH/src*所在的目录，执行以下命令来快速创建一个命名为*webplusdemo*的项目。

```
bee new webplusdemo
```

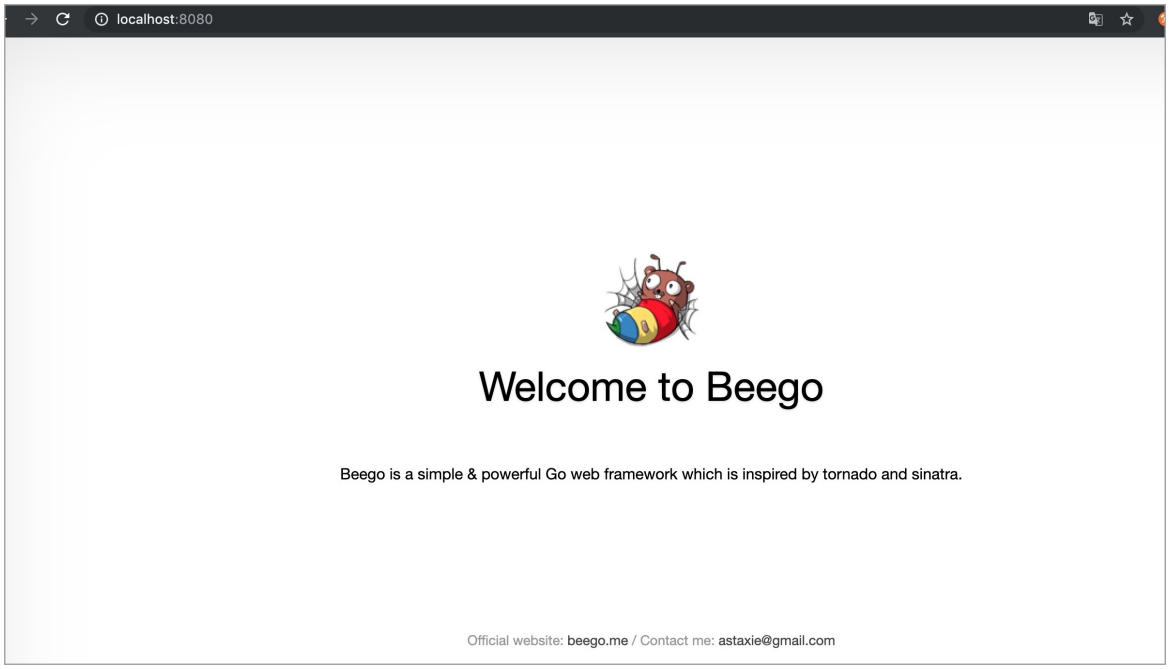
上述命令会创建一个名为*webplusdemo*的目录，结构如下：

```
webplusdemo
├── conf
│   └── app.conf
├── controllers
│   └── default.go
├── main.go
├── models
├── routers
│   └── router.go
├── static
│   ├── css
│   ├── img
│   └── js
│       └── reload.min.js
├── tests
│   └── default_test.go
└── views
    └── index.tpl
```

2. 进入项目目录，执行**bee run**。

```
______
| ___ \
| |_/ /  ___   ___
| ___ \ / _ \ / _ \
| |_/ /|  __/|  __/
\____/  \___| \___| v1.10.0
2019/07/31 12:54:33 INFO     ▶ 0001 Using 'webplusdemo' as 'appname'
2019/07/31 12:54:33 INFO     ▶ 0002 Initializing watcher...
2019/07/31 12:54:35 SUCCESS  ▶ 0003 Built Successfully!
2019/07/31 12:54:35 INFO     ▶ 0004 Restarting 'webplusdemo'...
2019/07/31 12:54:35 SUCCESS  ▶ 0005 './webplusdemo' is running...
2019/07/31 12:54:35.976 [I] [asm_amd64.s:1337]  http server Running on http://:8080
```

3. 打开浏览器，输入http://localhost:8080来访问项目。



步骤三：打包应用

1. 在项目目录下新建Procfile来给Go应用指定启动命令，填写以下内容，并保存。

```
web: chmod +x webplusdemo; ./webplusdemo
```

2. 使用bee工具来对打包工程。

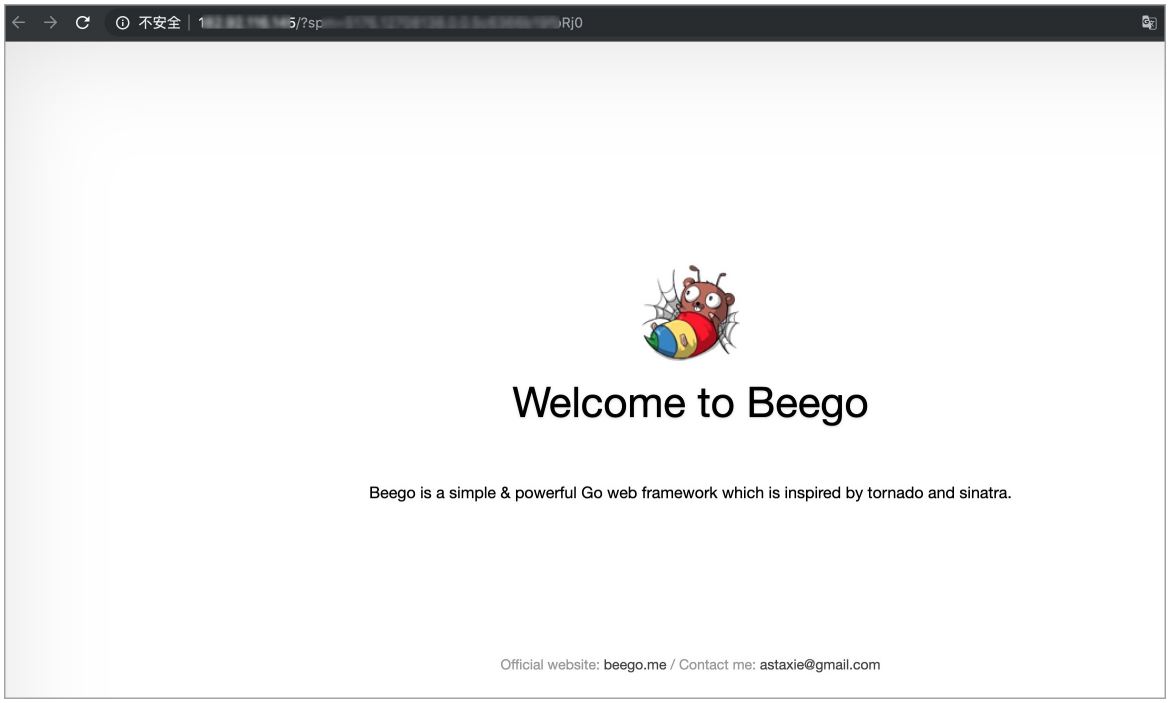
```
bee pack -be GOOS=linux -be GOARCH=amd64 -f zip
```

执行完上述命令，将会在项目目录下生成一个名为webplusdemo.zip的压缩包，按照下述步骤来将该Go应用部署至Web+。

步骤四：创建应用并完成部署

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为Go，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的webplusdemo.zip，设置部署包版本后单击**完成创建**。
5. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。



常见问题

健康检查失败导致网站无法访问怎么办？

如果使用SLB，请在健康检查URL的controller中接受**head**请求，否则可能会导致健康检查失败，网站无法访问。

```
package controllers
import (
 "github.com/astaxie/beego"
)
type MainController struct {
 beego.Controller
}
func (c *MainController) Get() {
 c.Data["Website"] = "beego.me"
 c.Data["Email"] = "astaxie@gmail.com"
 c.TplName = "index.tpl"
}
func (c *MainController) Head() {
 c.Ctx.Output.Body([]byte(""))
}
```

更多信息

- 在Web+控制台快速部署应用的视频演示请参见在Web+控制台创建应用和部署环境。
- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见使用CLI快速部署Java应用。
- 完成应用托管之后的应用的管理操作请参见应用详情概览。
- 管理应用所在的部署环境的操作请参见管理部署环境。

5.3. 向Go应用的部署环境中添加RDS实例

您可以使用阿里云云数据库RDS的数据库实例来存储应用中需持久保存的数据。本文将介绍如何为Go应用添加数据库并验证应用与数据库是否连接。

环境变量

Web+会将数据库连接的相关信息存放在环境变量中，以便应用进行读取，相关环境变量请参考下表。

变量名	变量值	变量说明
WP_RDS_ENGINE	MySQL	RDS数据库引擎
WP_RDS_CONNECTION_ADDRESS	rm-***.mysql.rds.aliyuncs.com	RDS内网连接地址
WP_RDS_PORT	3306	RDS端口号
WP_RDS_ACCOUNT_NAME	webplus	RDS数据库账号名称
WP_RDS_ACCOUNT_PASSWORD	*****	RDS账号密码
WP_RDS_DATABASE	webplus	RDS数据库

安装数据库驱动

执行以下命令，安装MySQL数据库驱动：

```
go get github.com/go-sql-driver/mysql
```

添加数据库

添加数据库的操作请参照以下样例代码进行配置。

```
package main
import (
 "database/sql"
 "fmt"
 _ "github.com/go-sql-driver/mysql"
 "os"
)
func main() {
 user := os.Getenv("WP_RDS_ACCOUNT_NAME")
 passwd := os.Getenv("WP_RDS_ACCOUNT_PASSWORD")
 host := os.Getenv("WP_RDS_CONNECTION_ADDRESS")
 port := os.Getenv("WP_RDS_PORT")
 connStr := fmt.Sprintf("%s:%s@tcp(%s:%s)/?timeout=30s", user, passwd, host, port)
 db, _ := sql.Open("mysql", connStr)
 defer db.Close()
 sqlTxt := "select 'OK' as result"
 rows, _ := db.Query(sqlTxt)
 var result string
 for rows.Next(){
  _ = rows.Scan(&result)
 }
 // output "OK"
 fmt.Println(result)
}
```

更多信息

关于如何使用Web+来管理RDS，可参考云数据库RDS。

6.PHP

6.1. 项目文件夹结构

PHP技术栈中使用ZIP包来部署应用时，ZIP包的项目结构内的*index.php*文件必须放在*public*文件夹内，否则会导致应用部署失败。

```
php-webapp
└─── public
  └─── index.php
```

6.2. 设置PHP开发环境

在本地开发环境测试PHP应用，需准备好相关的开发环境。本文将介绍PHP开发环境的设置步骤，并提供相关工具的安装页面链接。

安装PHP

请根据以下操作安装PHP和一些常用扩展。如果您没有特别的要求，请获取最新版本。

Linux

1. 在PHP官网下载安装包，例如*php-7.3.8.tar.bz2 (sig)*。
2. 进入安装包所在目录。
3. 执行以下命令安装包：

```
$ sudo yum install php
```

macOS

1. 在PHP官网下载安装包，例如*php-7.3.8.tar.bz2 (sig)*。
2. 执行以下命令安装包：

```
$ brew install php
```

Windows

1. 在PHP For Windows下载Windows系统安装包，例如*PHP 7.3 (7.3.8)*。
2. 进入PHP安装包所在目录，运行下载的文件即可安装，无需其他配置。

安装Composer

Composer 是用于PHP的依赖项管理器。您可以使用它来安装库、跟踪应用程序的依赖项并为热门PHP框架生成项目。

1. 使用来自getcomposer.org的PHP脚本安装Composer。

```
$ curl -s https://getcomposer.org/installer | php
```

2. 安装程序将在当前目录中生成PHAR文件。将此文件移动到环境PATH中的位置以便将此文件用作可执行文件。

```
$ mv composer.phar ~/.local/bin/composer
```

3. 使用require命令安装库。

```
$ composer require twig/twig
```

Composer 会将您在本地安装的库添加到您的项目 composer.json文件。在部署项目代码时，Web+将使用Composer在您的环境中的应用实例上安装此文件中列出的库。如果您在安装Composer时遇到问题，请访问Composer官网。

安装IDE

集成开发环境（IDE，Integrated Development Environment ）是用于提供应用开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面等工具，可以显著提高开发效率。以下是PHP开发中常用的IDE。

- Eclipse
- PhpStorm

6.3. 使用Laravel框架开发应用

Laravel是一套简洁、优雅的PHP Web开发框架。本文档将演示如何使用Laravel创建一个应用和添加MySQL数据库，并将其部署到Web+上。

前提条件

- 设置PHP开发环境。

> 注意 使用Laravel框架开发应用需要PHP 5.5.9或更高版本。

步骤一：创建应用

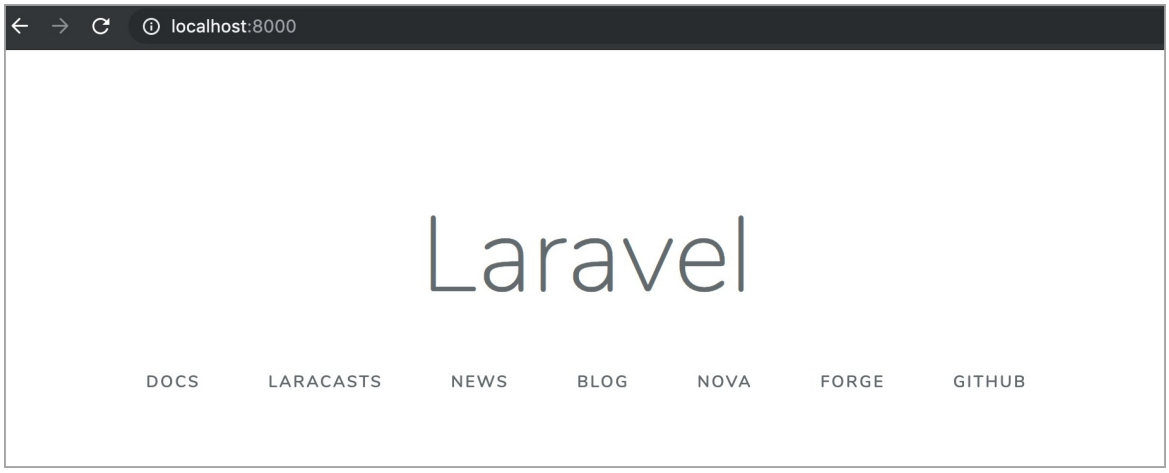
1. 执行以下命令使用Composer工具来创建一个名为webplusdemo的项目，该过程可能需要几分钟。

```
composer create-project --prefer-dist laravel/laravel webplusdemo
```

2. 执行以下命令使用PHP内置的开发服务器来运行此项目。

```
php artisan serve
```

3. 打开浏览器输入下图地址访问应用。



步骤二：打包应用

1. 进入项目目录，执行以下命令来激活Laravel内置的用户权限管理功能。

```
php artisan make:auth
```

2. 修改.env文件中数据库相关的配置，使其关联Web+的相关环境变量。

```
DB_CONNECTION=mysql
DB_HOST=${WP_RDS_CONNECTION_ADDRESS}
DB_PORT=${WP_RDS_PORT}
DB_DATABASE=${WP_RDS_DATABASE}
DB_USERNAME=${WP_RDS_ACCOUNT_NAME}
DB_PASSWORD=${WP_RDS_ACCOUNT_PASSWORD}
```

3. 执行以下命令，安装所有的依赖。

```
composer install
```

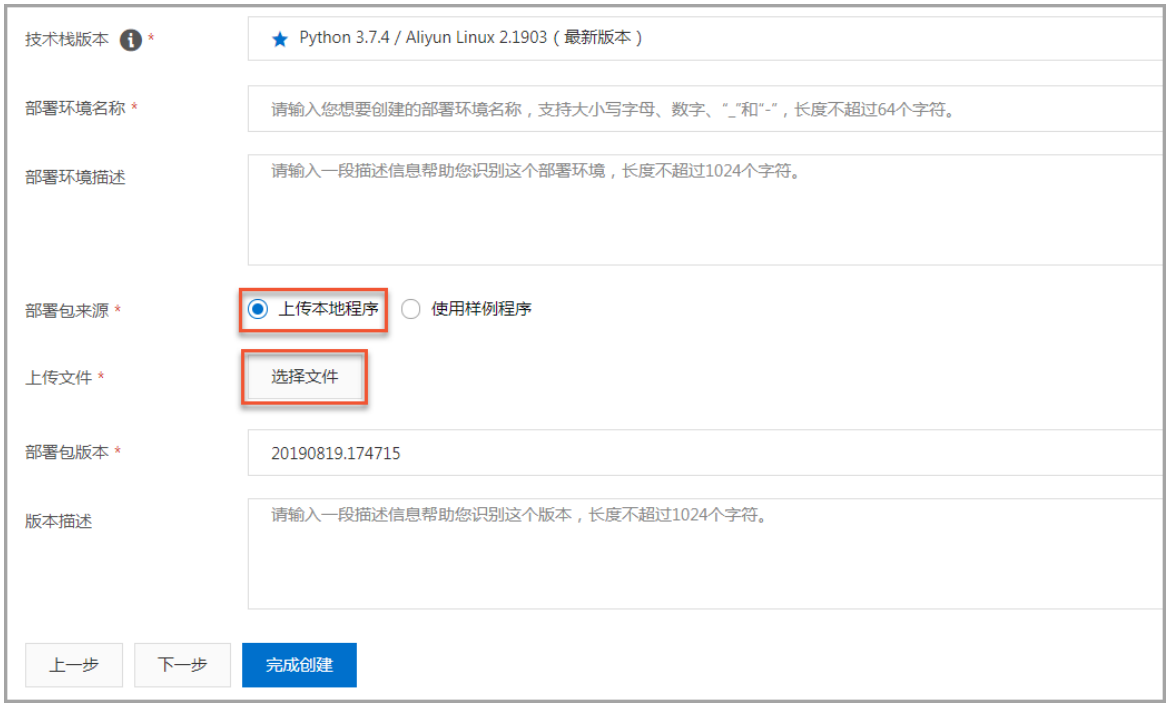
4. 使用**zip**命令打包项目下所有的内容生成压缩包**webplusdemo.zip**。

```
zip -r webplusdemo.zip ./
```

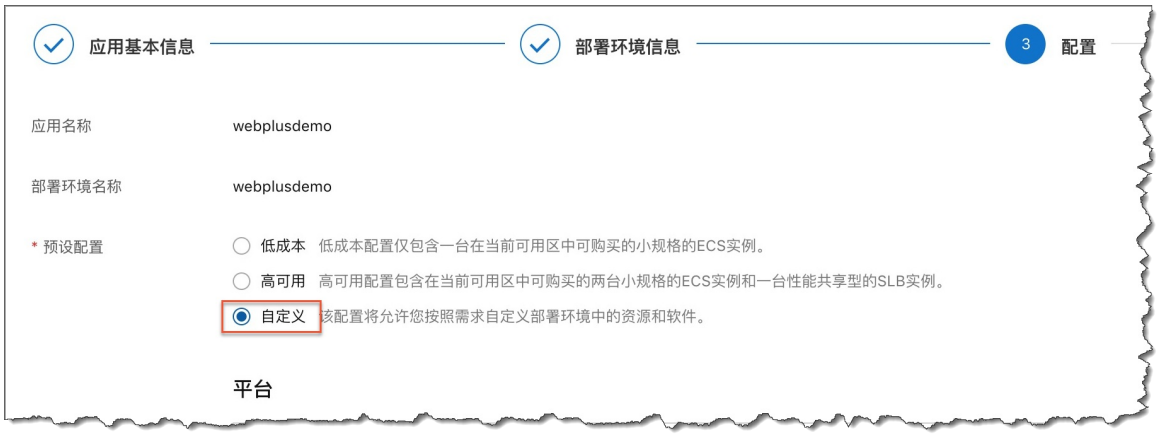
步骤三：部署应用至Web+

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为PHP，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的webplusdemo.zip，设置部署包版本后单击**下一步**。



5. 在**配置**页面选择**预设配置**为**自定义**模式。

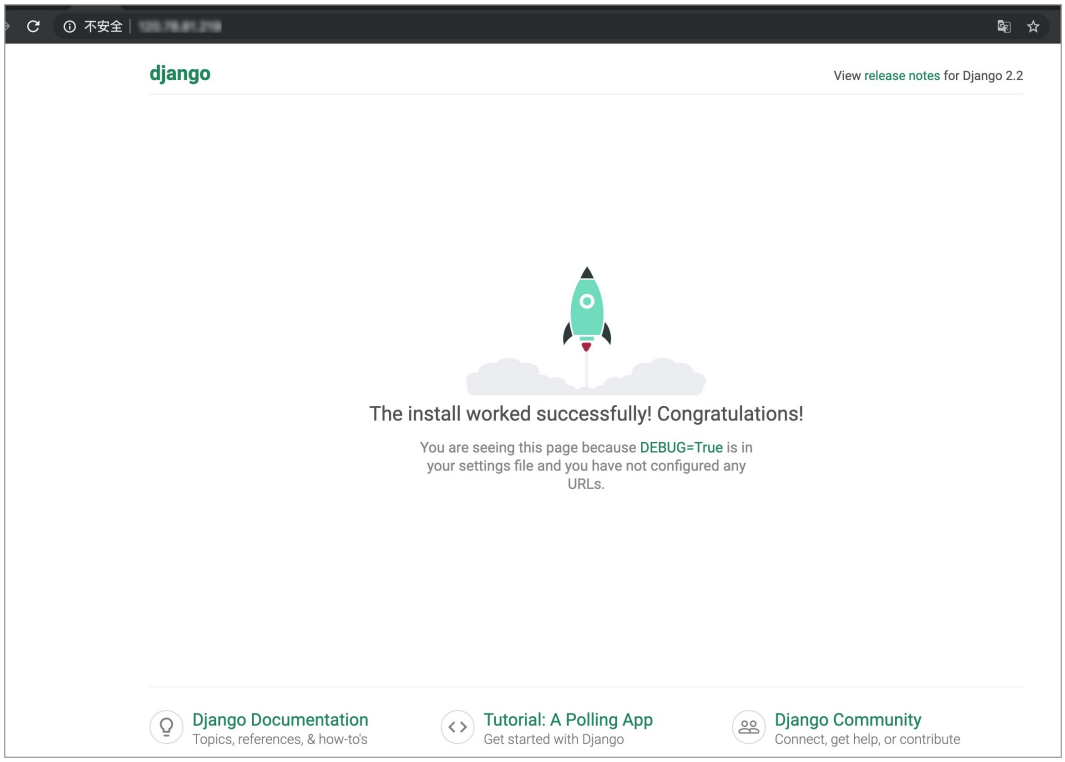


6. 展开**云数据库RDS**，按图所示配置云数据库类型为**MySQL**，并选择数据库版本、系列和类型等数据库基本信息。

7. 展开**生命周期挂钩**，在**PostPrepareApp**编辑框内输入以下内容。

```
cd $APP_HOME && /usr/local/php/bin/php artisan migrate
```

8. 在配置页面最下方单击**完成创建**。
9. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。



更多信息

- 关于如何使用Web+来管理RDS，可参考云数据库RDS。
- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见在CLI快速部署应用。

6.4. 使用Symfony开发应用

Symfony是一个基于MVC模式的面向对象的PHP框架，本文档将演示如何使用Symfony创建一个应用，并将其部署到Web+上。

前提条件

- 设置PHP开发环境。

> 注意 Symfony 3需要PHP 5.5.9或更高版本，以及PHP的intl扩展。

步骤一：安装Symfony

1. 执行以下命令安装Symfony CLI。

```
curl -sS https://get.symfony.com/cli/installer | bash
```

2. 将CLI的可执行文件移动到系统命令目录下。

```
mv ~/.symfony/bin/symfony /usr/local/bin/symfony
```

步骤二：创建应用

1. 执行以下命令使用CLI工具快速创建一个Symfony演示项目。

```
symfony new --demo webplusdemo
```

创建过程需要几分钟，创建完成后将生成名为webplusdemo的项目。

2. 进入项目目录执行以下命令安装依赖。

```
composer install
```

3. 执行以下命令，使用CLI工具附带的开发服务器启动服务。

```
symfony server:start
```

4. 打开浏览器访问http://localhost:8000，进入Symfony示例应用首页。

步骤三：打包应用

1. Web+支持您使用Nginx或Apache来作为Web服务器，若您希望使用Apache，可通过在项目目录下执行以下命令生成.htaccess文件，如果使用Nginx，则可跳过此步骤。

```
composer require symfony/apache-pack
```

2. 执行以下命令完成应用打包，生成部署包文件webplusdemo.zip。

```
zip -r webplusdemo.zip ./
```

步骤四：部署应用至Web+

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为PHP，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的webplusdemo.zip，设置部署包版本后单击**完成创建**。

5. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。

更多信息

- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见使用CLI快速部署Java应用。
- 想了解更多Symfony信息，请进入Symfony官方网站查看。

7.Python

7.1. 设置Python开发环境

在本地开发环境测试Python应用，需准备好相关的开发环境。本文将介绍Python开发环境的设置步骤，并提供相关工具的安装页面链接。

安装Python

根据您的开发平台，从Python官网下载安装包。

> 说明 为了和Web+的技术栈版本更好兼容，建议您下载Python 3.7.4或Python 2.7.16。

Linux

1. 进入Python安装包（例如*Python-3.7.4.tgz*）所在目录，执行以下命令解压安装包。

   ```
   tar xvf Python-3.7.4.tgz
   ```

2. 进入Python安装包所在目录，执行以下命令进行编译和安装。

   ```
   ./configure --with-ensurepip=install
   make && make install
   ```

macOS

执行以下命令来使用**brew**来快速安装Python。

```
brew update && brew install python
```

Windows

进入Python安装包所在目录，双击安装包按照提示指令进行安装。

虚拟环境

当同时开发多个项目时，使用Python虚拟环境来隔离这些项目是比较好的实践，这可以让您避免项目之间的依赖包的版本可能冲突的问题。

虚拟环境的具体用法请参见虚拟环境和包。

使用IDE

使用集成开发环境 (IDE) 提供了便于应用程序开发的大量功能，可以显著提高开发效率，以下是常见于Python开发中使用的IDE：

- Eclipse
- PyCharm（商业软件）

7.2. 使用Flask开发应用

Flask是Python的一个轻量级Web应用框架。本文档将演示如何使用Flask创建一个应用，并将其部署到Web+上。

步骤一：创建应用

1. 按照以下目录结构创建一个名为webplusdemo的目录，并在该目录下创建一个名为application.py的文件。

```
webplusdemo
└──── application.py
```

2. 在application.py文件中输入以下内容。

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
  return 'Hello World!'
```

> 注意
>
> - Web+会自动识别名为application.py的文件和全局变量app，若您需要使用其他文件名，需要通过Procfile或启动命令来指定需要的启动命令，具体操作请参见使用Procfile配置应用进程和命令与生命周期挂钩。
> - Web+默认会使用gunicorn作为Flask应用的服务器。

步骤二：打包应用

进入webplusdemo目录，执行以下命令完成应用打包，生成部署包文件webplusdemo.zip。

```
zip -r webplusdemo.zip ./
```

步骤三：部署应用至Web+

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为**Python**，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的webplusdemo.zip，设置部署包版本后单击**完成创建**。

5. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。

更多信息

- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见使用CLI快速部署Java应用。
- 想了解更多Flask信息，请进入Flask官方网站查看。

7.3. 使用Django开发应用

Django是Python的一个开放源代码的Web应用框架。本文档将演示如何使用Django创建一个应用和给应用添加MySQL数据库，并将其部署到Web+上。

步骤一：安装Django

执行以下命令安装Django。由于在本文档中将使用MySQL，因此需要安装pymysql模块。

```
pip install Django pymysql
```

> **注意** 使用Django（2.2版本以上）需要Python 3.5以上版本，在本文档将使用3.7.4版本作为示例。

步骤二：创建应用

1. 执行以下命令执行**django-admin**命令来快速创建一个项目。

   ```
   django-admin startproject webplusdemo
   ```

2. 可以看到创建的目录结构如下。

   ```
   webplusdemo/
   ├── manage.py
   └── webplusdemo
       ├── __init__.py
       ├── settings.py
       ├── urls.py
       └── wsgi.py
   ```

3. 执行以下命令执行**django-admin**命令来快速创建一个项目。

   ```
   django-admin startproject webplusdemo
   ```

步骤三：打包应用

1. 执行以下命令来修改settings.py中**ALLOWED_HOSTS**配置项，允许所有域名的访问。

   ```
   ALLOWED_HOSTS = ['*']
   ```

2. 执行以下命令改写settings.py中的数据库配置。Django默认使用sqlite数据库，本示例中将使用RDS中的MySQL数据库。

```
# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': os.environ['WP_RDS_DATABASE'],
    'USER': os.environ['WP_RDS_ACCOUNT_NAME'],
    'PASSWORD': os.environ['WP_RDS_ACCOUNT_PASSWORD'],
    'HOST': os.environ['WP_RDS_CONNECTION_ADDRESS'],
    'PORT': os.environ['WP_RDS_PORT'],
  }
}
```

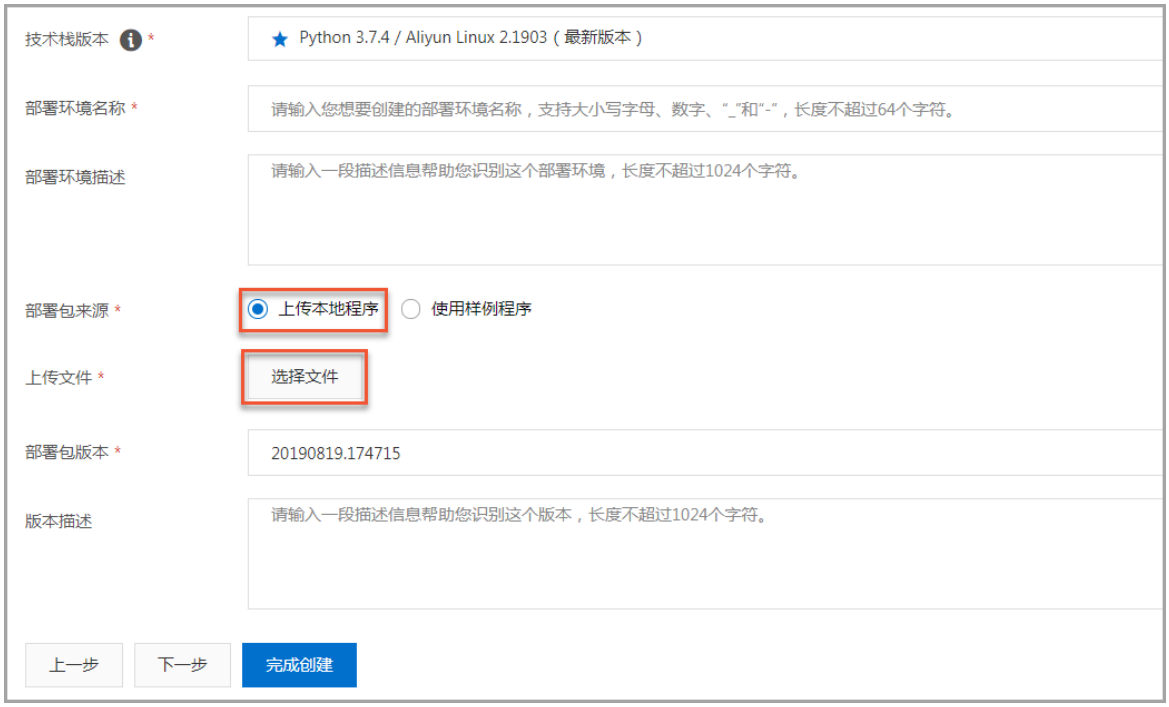
3. 进入webplusdemo目录，执行以下命令完成应用打包，生成部署包文件webplusdemo.zip。

```
zip -r webplusdemo.zip ./
```

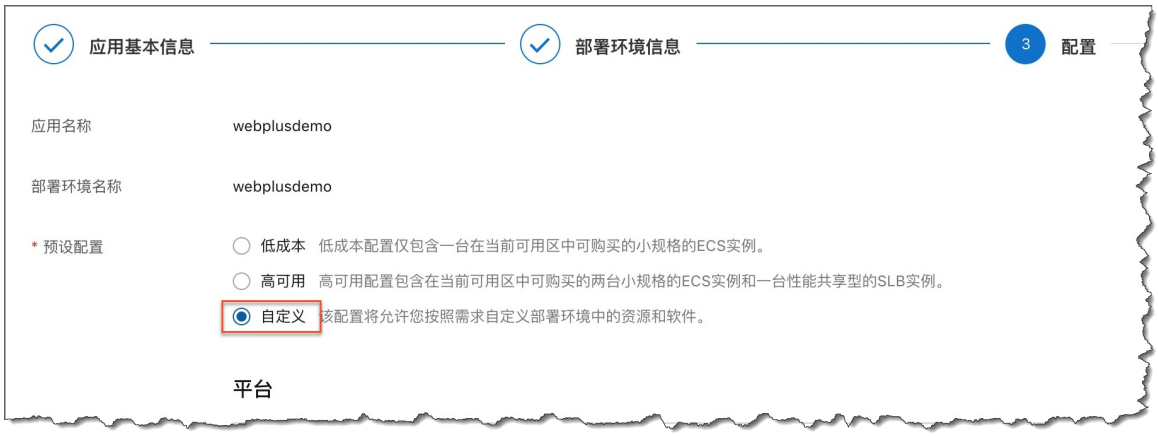
步骤四：部署应用至Web+

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为**Python**，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的webplusdemo.zip，设置部署包版本后单击**下一步**。



5. 在**配置**页面选择**预设配置**为**自定义**模式。

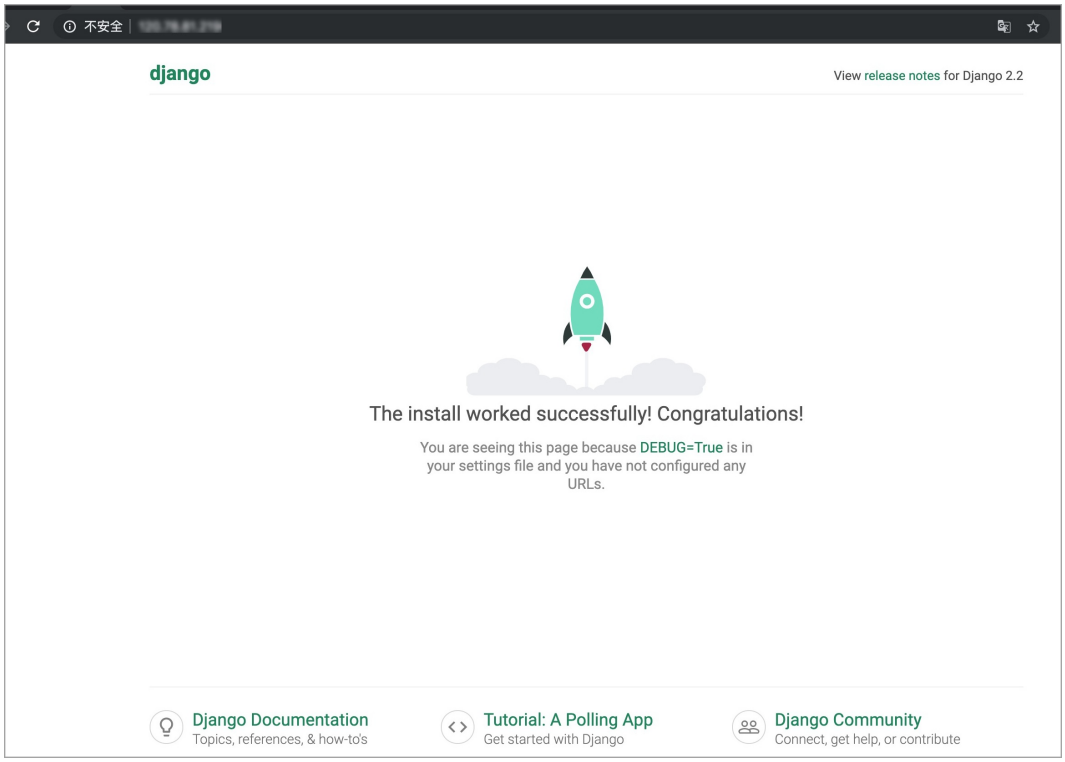


6. 展开**云数据库RDS**，按图所示配置云数据库类型为**MySQL**，并选择数据库版本、系列和类型等数据库基本信息。

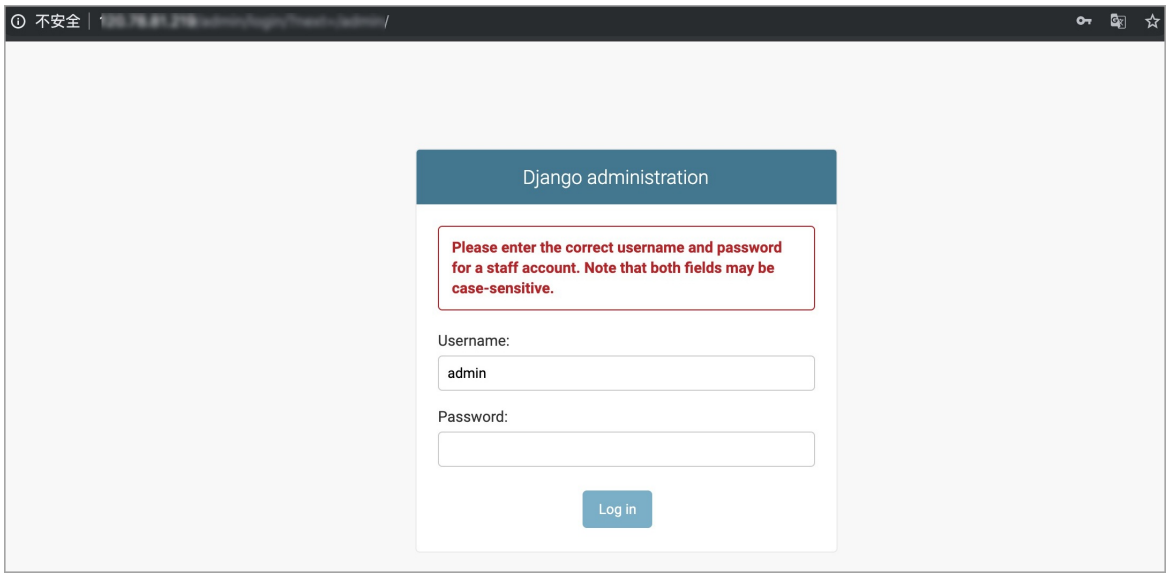
7. 展开**生命周期挂钩**，在**PostPrepareApp**编辑框内输入以下内容。配置完成后单击**完成创建**。

```
source /etc/bashrc && cd $APP_HOME && python manage.py migrate
```

8. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。



上面步骤配置了数据库，因此可以访问登录页。您可以通过在数据库写入用户表或执行 `python manage.py createsupersuer` 的方式来创建用户。



更多信息

- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见在CLI快速部署应用。
- 想了解更多Django信息，请进入Django官方网站或Django Github项目查看。

8.ASP.NET Core

8.1. 设置ASP.NET Core开发环境

在本地开发环境测试ASP.NET Core应用，需要准备相关的开发环境。本文将介绍ASP.NET Core开发环境的设置步骤，并提供相关工具的安装页面链接。

安装Visual Studio

对于开发ASP.NET Core应用，Visual Studio系列工具提供了很多便捷的功能，推荐使用macOS或者Windows的用户在Visual Studio官网下载并安装Visual Studio。

安装SDK

使用Linux的开发者需要安装.NET Core SDK，以在CentOS 7系统中安装.NET Core 2.2 SDK为例。

执行以下命令来安装SDK。 或参考Install .NET Core 2.2 SDK on Linux CentOS 7 - x64文档来完成安装。

```
sudo rpm -Uvh https://packages.microsoft.com/config/centos/7/packages-microsoft-prod.rpm
sudo yum update
sudo yum install dotnet-sdk-2.2
```

8.2. 部署ASP.NET Core应用至Web+

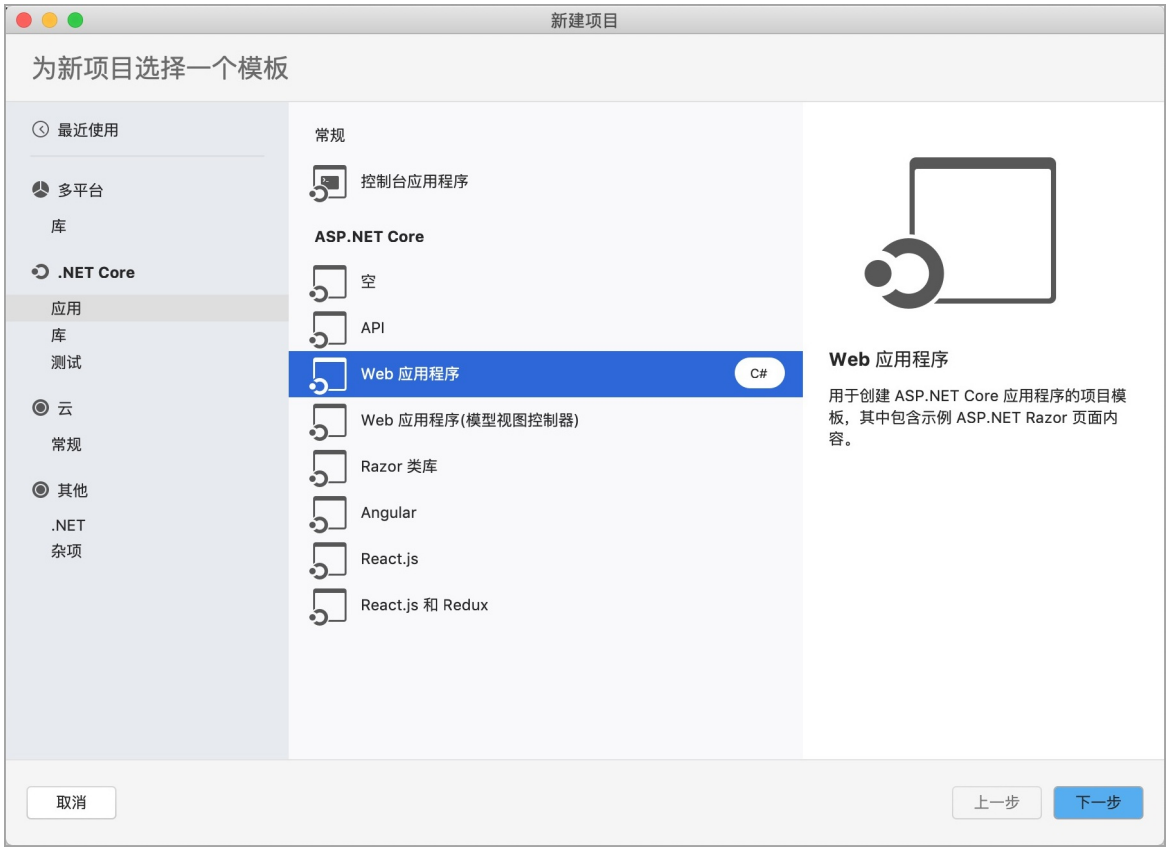
Razor 页面是 ASP.NET Core MVC 的一个新功能，它可以使基于页面的编码方式更简单高效。本文将以ASP.NET Core Razor页面Web应用为例，介绍如何创建一个ASP.NET Core 应用并将其部署到Web+。

前提条件

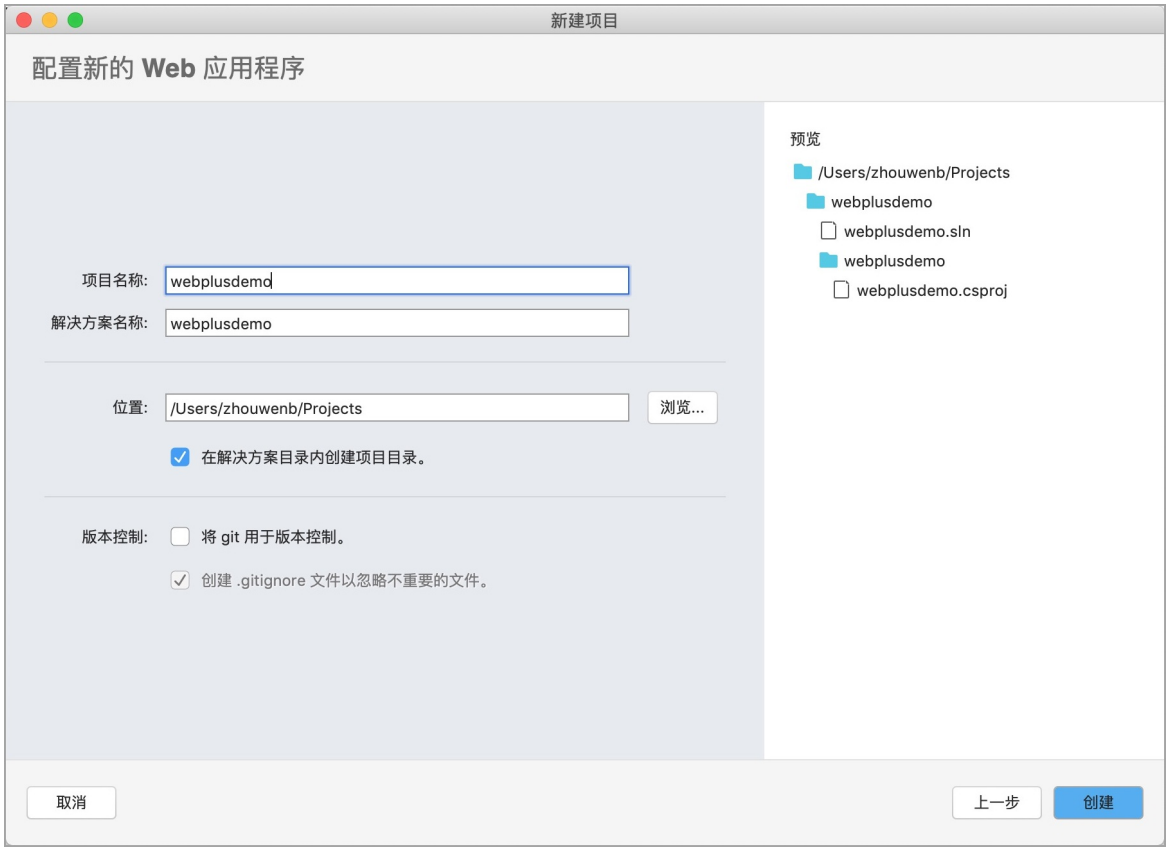
设置ASP.NET Core开发环境

步骤一：使用Visual Studio创建应用

1. 使用Visual Studio新建一个项目，选择**Web应用程序**。

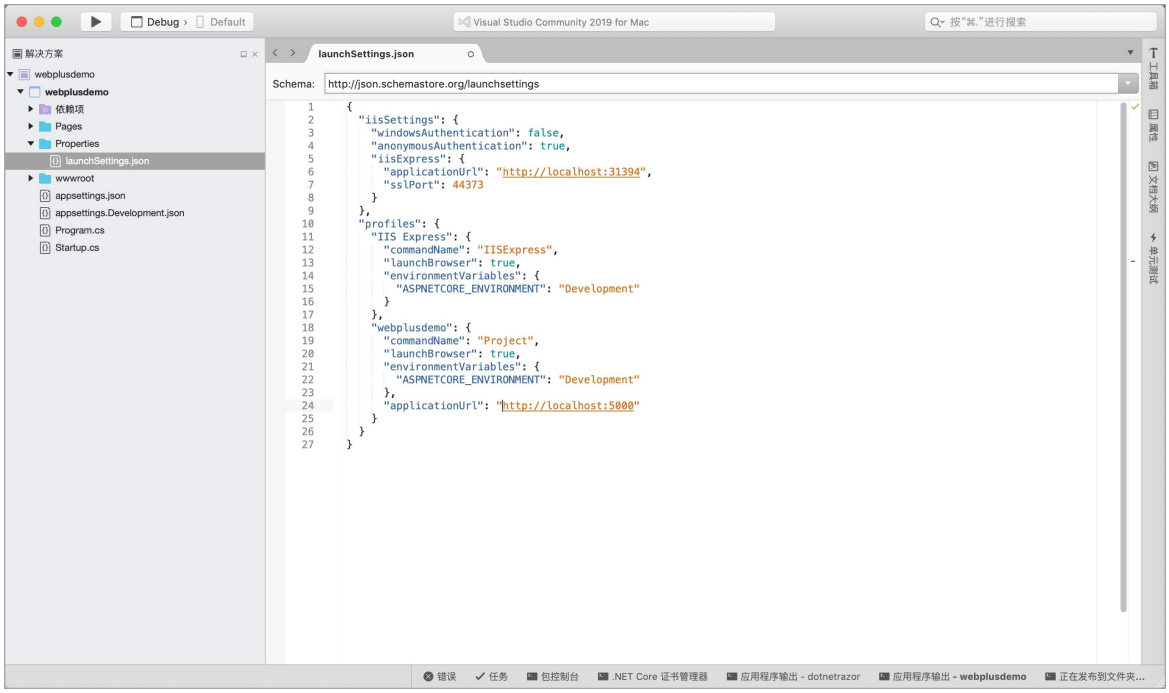


2. 在**配置新的Web应用程序**页面填写项目名称为webplusdemo，然后单击**创建**。

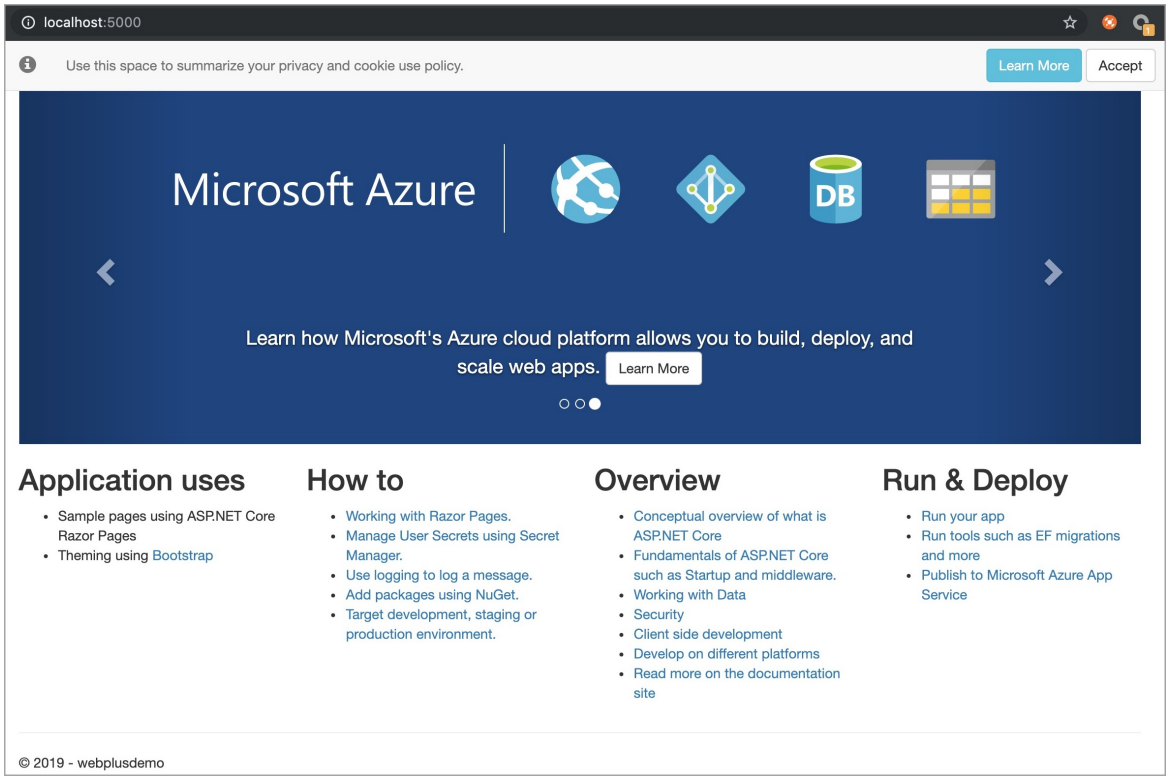


3. Web+可以在反向代理或SLB中配置HTTPS，在此处您可以将应用自身的HTTPS URL去掉，请按以下示例

修改Properties/launchSettings.json文件中的 `applicationURL` 配置，仅保留HTTP的访问链接。

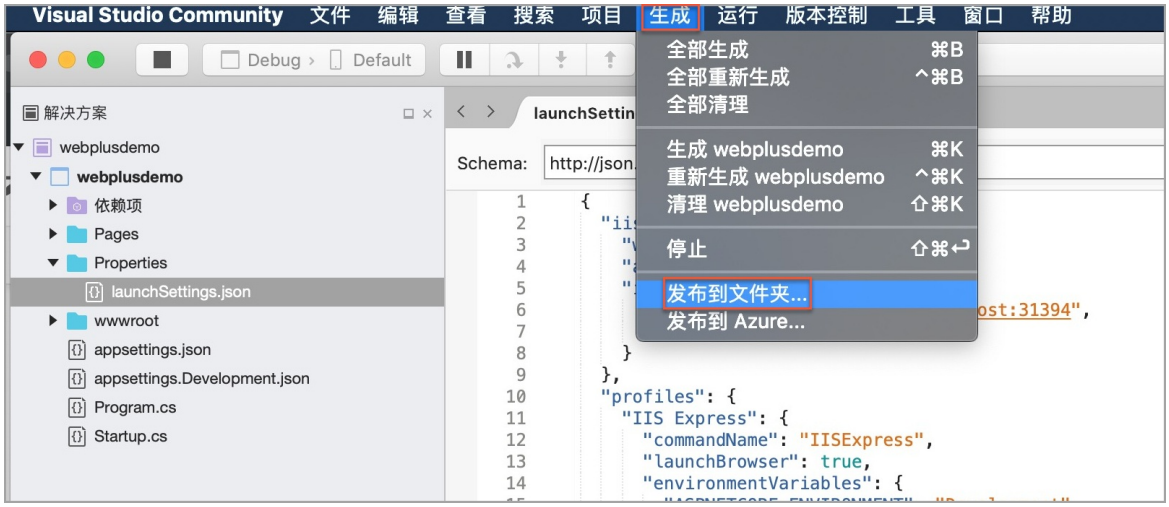


4. 在浏览器地址栏输入http://localhost:5000/来访问项目。



步骤二：打包应用

1. 在Visual Studio顶部菜单栏选择**生成 > 发布到文件夹**，并指定一个目标路径。



2. 通过终端访问部署包的目标路径，可以看到已经成功生成了文件。

3. 执行以下命令完成打包，在当前目录下得到部署包文件webplusdemo.zip。

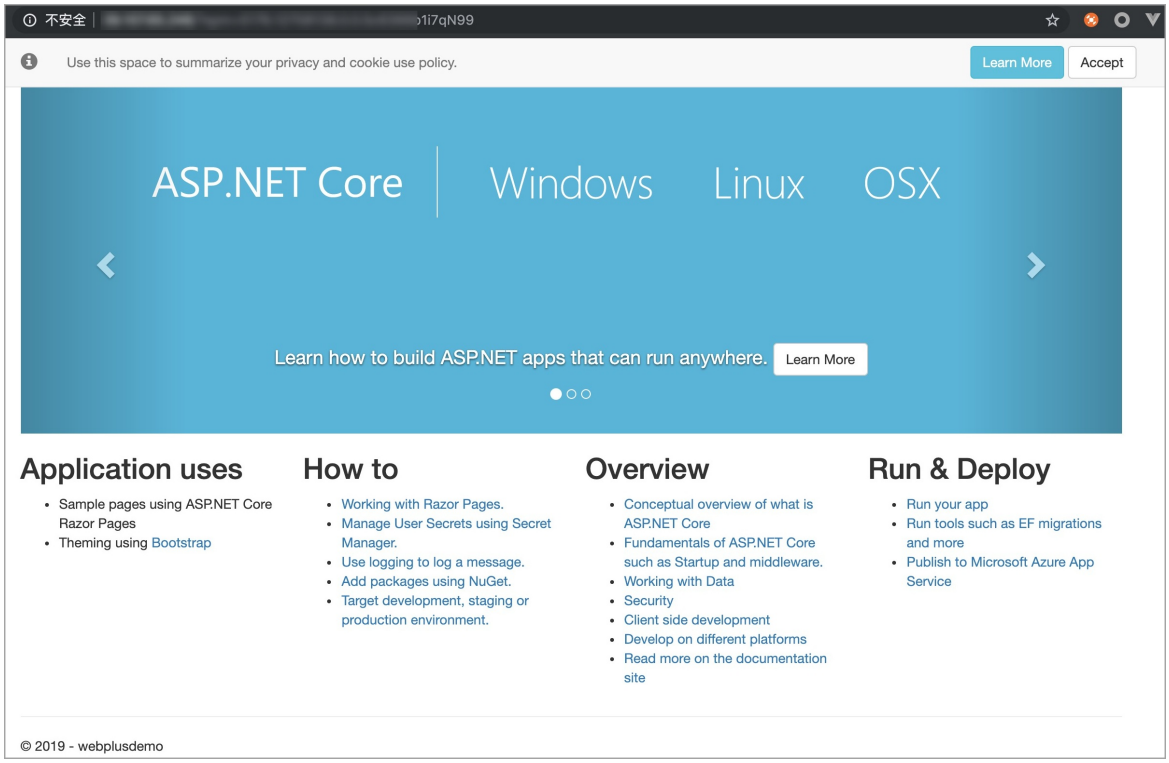
```
zip -r webplusdemo.zip ./
```

步骤三：将应用部署至Web+

1. 登录Web+控制台，并在页面左上角选择所属地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**创建**。
3. 在**应用基本信息**页面选择**技术栈类型**为**ASP.NET Core**，设置应用基本信息，设置完成后单击**下一步**。

参数	描述
技术栈类型	根据您的实际业务需求选择技术栈类型，在本教程中选择**ASP.NET Core**。
应用名称	设置应用名称，此处以demo作为示例。
应用描述	输入创建应用的描述信息，可以选择不设置。
使用共享存储空间	使用共享存储空间是指在应用生命周期的整个过程中，上传的部署包、采集的日志和诊断信息等都将存储在该空间中，降低了产品的使用成本。如果您对数据隐私有较高要求，可以选择关闭该功能，这时所有数据将存储在您自己的OSS空间内。

4. 在**部署环境信息**页面设置环境和部署包信息，部署包来源选择**上传本地程序**，上传您刚打包的部署包，设置部署包版本后单击**用低成本预设配置创建**。
5. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。



更多信息

- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见使用CLI快速部署Java应用。
- .NET Core相关操作指南，请参见.NET Core 指南。

9.Ruby

9.1. 配置Ruby开发环境

在本地开发环境测试Ruby应用，需要准备相关的开发环境。本文将介绍Ruby开发环境的设置步骤，并提供相关工具的安装页面链接。

安装 Ruby

您可以参考Ruby官网文档来安装Ruby，建议您安装Web+支持的版本Ruby 2.6.3。

Linux

Linux下使用系统包管理是安装Ruby最简单的方法，以Centos 7为例，执行以下命令即可完成安装：

```
yum install ruby
```

若您希望安装最新版本或指定版本的Ruby，则需要下载对应的源代码版本进行编译安装，下载解压后执行以下命令：

```
./configure
make && make install
```

macOS

执行以下命令，使用brew来快速安装Ruby。

```
brew update && brew install ruby
```

Windows

如果您使用Windows系统，可以借助RubyInstaller来完成Ruby的安装。

配置GEM

Gem为Ruby的包管理工具，中国内地网络访问默认Gem源的速度偏慢，可以通过使用镜像站点来加速访问，安装好Ruby后请执行以下命令来配置GEM。

```
gem sources --add https://gems.ruby-china.com/ --remove https://rubygems.org/
bundle config mirror.https://rubygems.org https://gems.ruby-china.com
```

安装IDE

集成开发环境 (IDE) 提供了便于应用程序开发的大量功能，可以显著提高开发效率。RubyMine是常见在Ruby开发中使用的IDE，请访问链接下载软件安装包并安装：RubyMine（商业软件）。

9.2. 部署Ruby on Rails应用至Web+

Ruby on Rails（后文简称Rails）是一个使用Ruby语言开发的开源Web应用框架，本文将介绍如何创建一个简单的Rails项目并将其部署到Web+。

步骤一：安装Rails

执行以下命令安装Rails。

```
gem install rails
```

步骤二：创建Rails应用

1. 执行以下命令使用rails命令行工具来创建一个空应用。

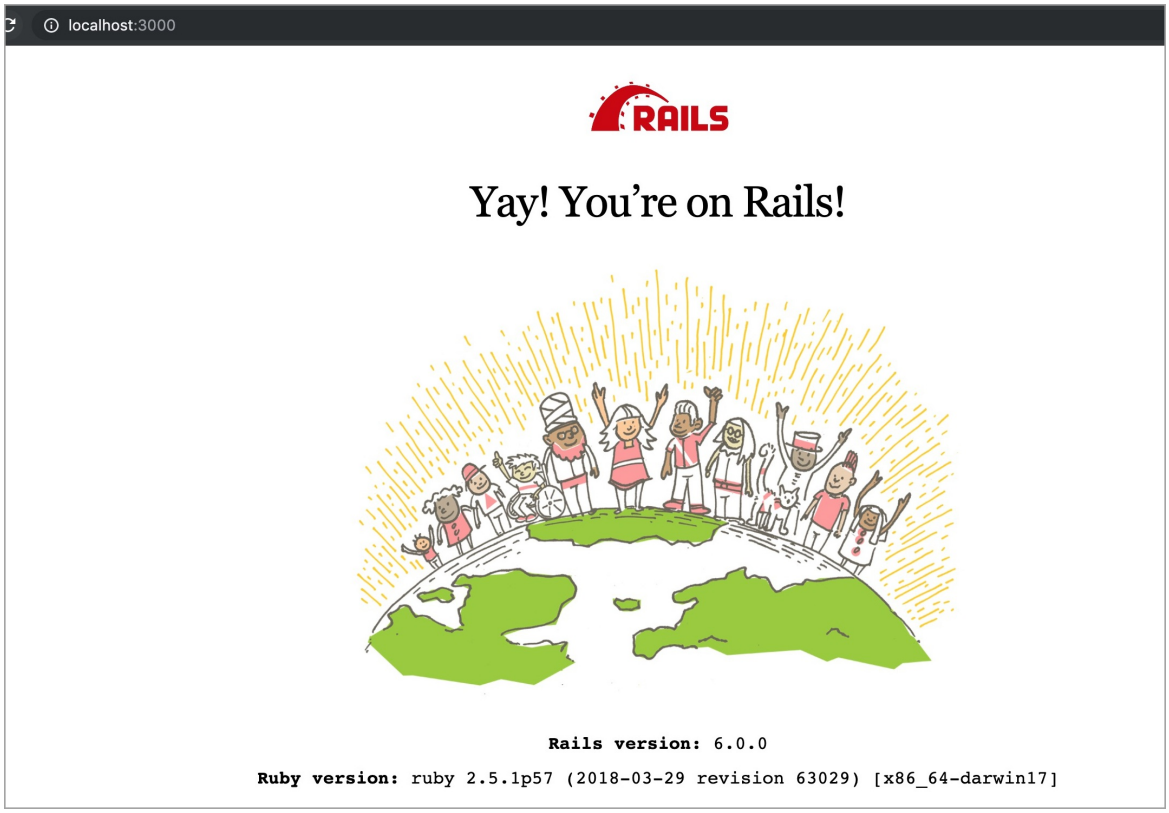
   ```
   rails new webplusdemo
   ```

   执行完后将生成名为webplusdemo的工程目录。

2. 进入webplusdemo的工程目录，执行以下命令来启动服务。

   ```
   rails server
   ```

3. 打开浏览器，输入http://localhost:3000来访问应用首页。



步骤三：打包应用

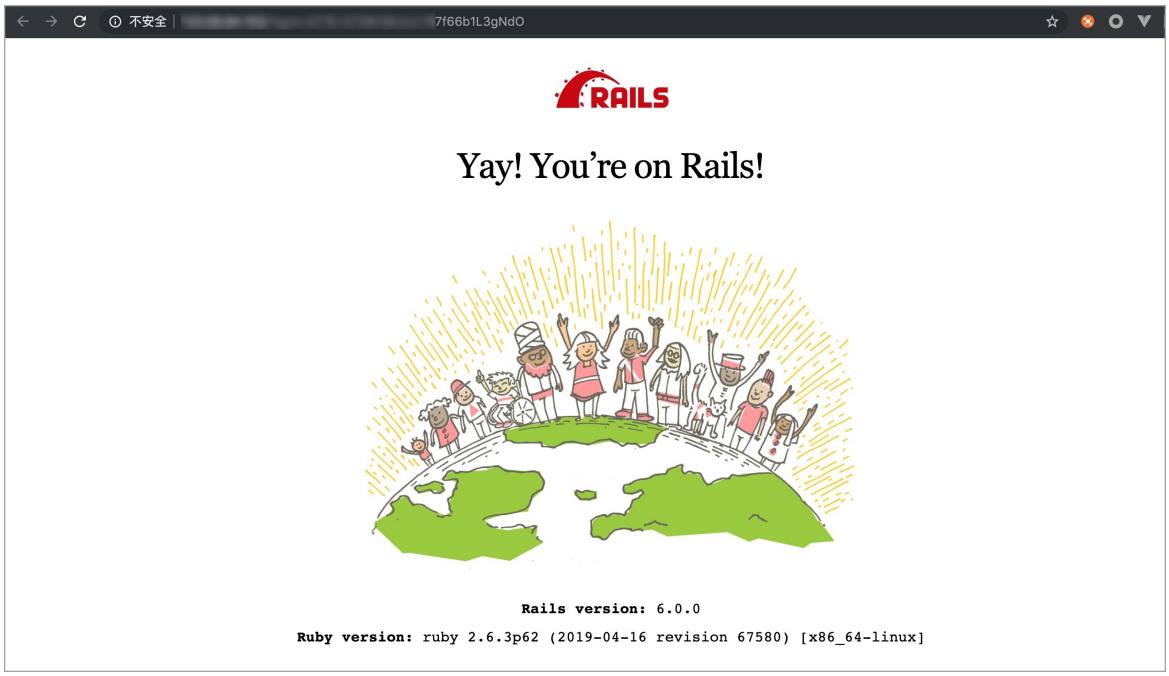
切换到项目路径下执行以下命令完成打包。

```
zip -r webplusdemo.zip ./
```

步骤四：部署Ruby应用至Web+

1. 登录Web+控制台，并在页面左上角选择所属地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**创建**。
3. 在**应用基本信息**页面选择**技术栈类型**为**Ruby**，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的部署包，设置部署包版本后单击**完成创建**。
5. 在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。



更多信息

- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见使用CLI快速部署Java应用。
- 若想详细了解 Ruby on Rails，请访问Rails官网。

10.Native

10.1. 部署原生应用到Web+

Web+提供了丰富的多语言应用的支持，如果您无法找到适合的技术栈类型，您可以选择原生应用来完成部署。本文将介绍原生应用部署到Web+的设置步骤。

原生应用简介说明

- Web+不会自动安装基础软件，您可通过命令与生命周期挂钩来自定义安装步骤，比如在Post PrepareEnv挂钩中使用命令安装所需软件或依赖。
- Web+不提供默认的启动命令，因此必须通过配置命令或参考文档使用Procfile配置应用进程来指定启动命令。
- Web+默认原生应用的服务端口为8080，若您的应用没有启动在此端口，请将服务端口配置到实际启动的端口，您可使用环境变量**$WP_SERVICE_PORT**来配置应用的服务端口。如果应用的服务端口跟启动端口不一致可能导致应用健康检查失败。
- 您的应用必须是在对应的部署环境中（目前操作系统只支持*AliyunLinux2.1903*）下可以正常运行的。
- 原生应用也可以使用数据库，配置方法请参考云数据库RDS。您可以从环境变量读取到Web+设置的数据库选项，选择合适的驱动程序即可实现数据库访问，环境变量请参考环境变量。

打包原生应用

本文以一个简单的HTTP服务程序*simpleserver*作为示例，该程序只包含一个可执行文件，启动后将监听在8080端口，并接收GET请求，输出OK的响应。

```
.
└── simpleserver
```

1. 在应用项目目录下创建Procfile，文件中写入以下内容：

   ```
   web: ./simpleserver
   ```

2. 执行以下命令打包应用，即可生成可在Web+使用的部署包。

   ```
   zip -r simpleserver.zip ./
   ```

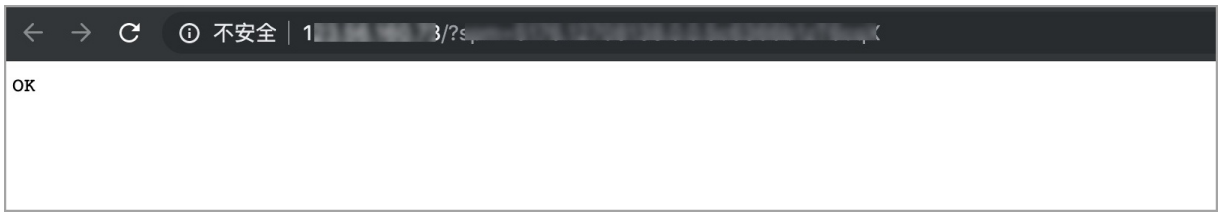
将原生应用部署至Web+

1. 登录 Web+控制台，并在页面左上角选择所需地域。
2. 在**概览**页**最近更新的部署环境**区域的右上角单击**新建**。
3. 在**应用基本信息**页面选择**技术栈类型**为**Native**，设置应用基本信息，设置完成后单击**下一步**。

4. 在**部署环境信息**页面设置**部署环境名称**，部署包来源选择**上传本地程序**，上传您刚打包的部署包，设置部署包版本后单击**完成创建**。

访问应用

在**完成创建**页面单击**查看该应用**或**完成创建**可进入**应用详情**页面。单击部署环境名称进入**部署环境详情**页面，然后单击**公网访问地址**右侧的链接进入应用首页。



更多信息

- 在Web+控制台快速部署应用的视频演示请参见在Web+控制台创建应用和部署环境。
- 在控制台部署应用的详细配置步骤请参见部署应用。
- 使用CLI完成应用创建和部署的操作请参见使用CLI快速部署Java应用。
- 完成应用托管之后的应用的管理操作请参见应用详情概览。
- 管理应用所在的部署环境的操作请参见管理部署环境。