阿里云
轻量级分布式应用服务

应用部署

文档版本：20190909

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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	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。	`swich` *`{stand | slave}`*

目录

1 应用部署概述

SAE 是面向应用的 Serverless PaaS 平台，向上抽象了应用的概念，您可以无需管理和维护集群与服务器，只需专注于设计和构建应用程序，将其部署在 SAE。应用开发完成以后，如果首次部署，需要创建并部署应用；如果已经部署在 SAE 上，可以使用部署应用进行版本升级。

部署方式

通过控制台或开发工具两种方式进行部署。

升级时，建议使用灰度发布或者分批发布。

部署应用的方式可参考以下表格。

应用举例	部署方式	参考开发文档
原生 Spring Cloud	WAR、JAR、镜像	将 Spring Cloud 应用托管到 SAE
原生 Dubbo	WAR、JAR、镜像	将 Dubbo 应用托管到 SAE
HSF	WAR、JAR、镜像	/
多语言应用	镜像	/

使用Spring Cloud、Dubbo 和 HSF 框架来创建的应用都可以部署在 SAE 中，所选择部署方式不同则应用的运行环境不同。

- Spring Cloud 和 Dubbo 应用通过 WAR 包部署时，选择 `apache-tomcat` 相关版本的运行环境。
- Spring Cloud 和 Dubbo 应用通过 JAR 包部署时，选择 标准 `Java` 应用运行环境。
- HSF 应用通过 WAR 或 JAR 包部署时，选择 `EDAS-Container` 相关版本的运行环境。

控制台部署

使用控制台部署应用 SAE。建议使用 Chrome 浏览器进行控制台操作。

工具部署

除通过控制台方式进行应用部署，还可通过以下工具进行部署。

应用部署高级设置

在应用部署时，可以参考以下文档进行启动命令、环境变量、Host绑定、健康检查和设置日志收集等。

2 控制台部署

2.1 在 SAE 控制台使用 War 包部署 Java Web 应用

应用开发完成后，可以将您的应用部署到 SAE 进行托管。本文介绍如何在 SAE 控制台使用 War 包部署 Java Web 应用。

在 SAE 中部署应用的方式如下表所示。

应用举例	部署方式	参考开发文档
原生 Spring Cloud	WAR、JAR、镜像	将 Spring Cloud 应用托管到 SAE
原生 Dubbo	WAR、JAR、镜像	将 Dubbo 应用托管到 SAE
HSF	WAR、JAR、镜像	/
多语言应用	镜像	/

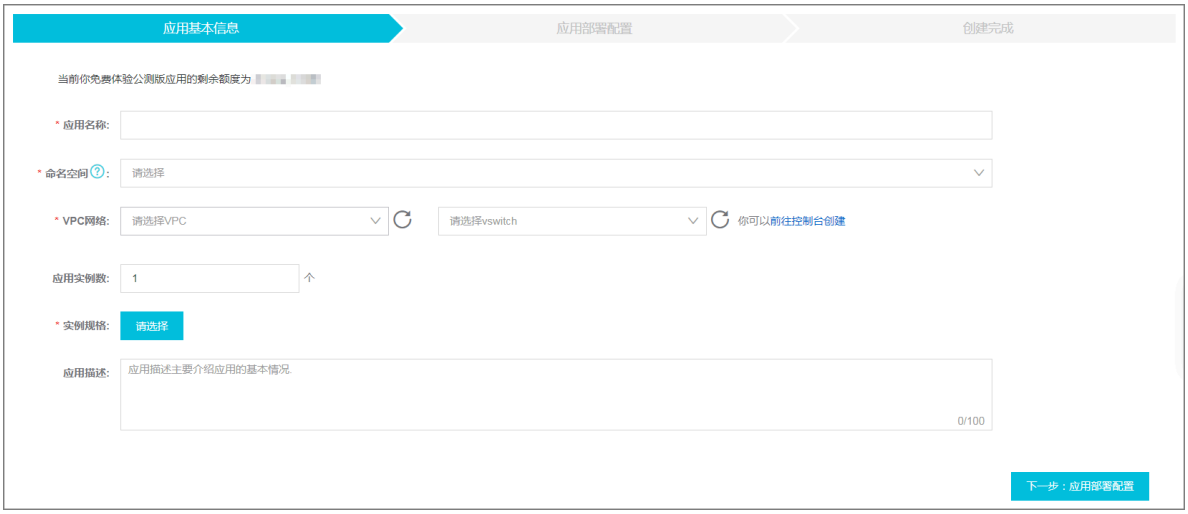
前提条件

- #unique_6/unique_6_Connect_42_section_cu5_k9p_xuf。
- 有环境隔离需求，请#unique_6/unique_6_Connect_42_section_xrz_zr9_py3。

创建 SAE 应用

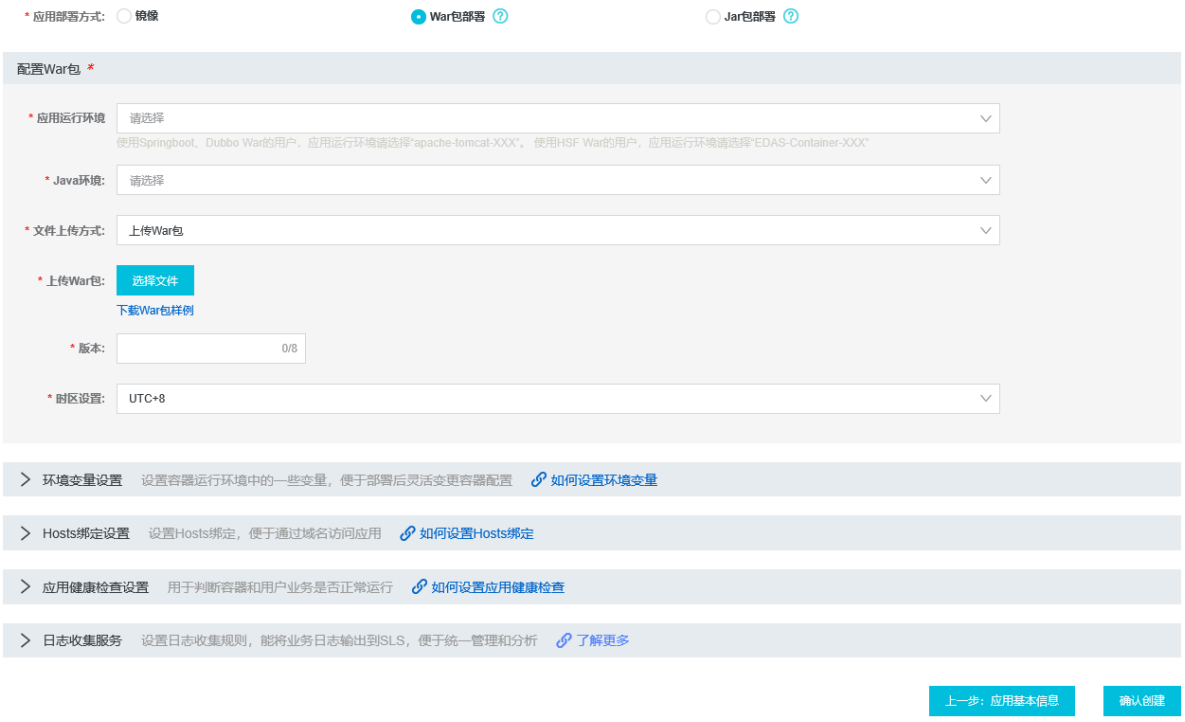
1. 登录 SAE 控制台。
2. 在左侧导航树选择Serverless 应用引擎 > 应用列表，并在 应用列表页面单击右上角 创建应用。

3. 在 应用基本信息页签内，设置应用相关信息，配置完成后单击 下一步：应用部署配置。



- 应用名称：输入应用名称。允许数字，字母，下划线以及中划线组合，仅允许字母开头，最大长度 36 个字符。
- 命名空间：在下拉菜单中选择创建好的命名空间。
- VPC 网络：在下拉菜单中选择VPC 和 vswitch 。
- 应用实例数：选择要创建的实例个数。
- 实例规格：单击请选择，在选择实例规格页面内选择实例的 CPU 和 Memory 规格。
- 应用描述：填写应用的基本情况，输入的描述信息不超过100个字符。

4. 在应用部署配置页面，选择应用部署方式选择 War 包部署，依据页面指示进行配置。完成设置后单击确认创建。



- 应用运行环境：使用 Springboot 或 Dubbo 的用户，应用运行环境请选择*apache-tomcat-XXX*；使用 HSF 的用户，应用运行环境请选择*EDAS-Container-XXX*。
- Java环境：选择 *Open JDK 7* 或 *Open JDK 8*。
- 文件上传方式：可选择*## War #*或*War ###*。
  - 上传 War 包：单击选择文件，选择待部署 War 包。
  - War 包地址：输入 War 包的存放地址。

  > **注意：**
  > 应用部署程序包名仅允许字母、数字，及中划线“-”、下划线“_”两个特殊符号。
- 版本：设置版本（如：1.1.0），不建议用时间戳作为版本号。
- 时区设置：选择当前应用所在时区，如*UTC+8*。
- 环境变量设置（可选）：请参见如何设置环境变量配置。
- Hosts 绑定设置（可选）：请参见如何设置 Hosts 绑定配置。
- 应用健康检查（可选）：请参见如何设置应用健康检查配置。
- 日志收集设置（可选）：请参见如何设置日志收集配置。

5. 进入应用详情页，查看应用的基本信息和实例部署信息。

结果验证

在应用详情页的实例部署信息页签查看实例的运行状态。如果运行状态显示为绿色的 Running或者 Completed，表示应用部署成功。

更多信息

- 使用 SAE 创建应用后，可以进行更新、扩缩容、启停应用监控和删除应用等管理操作，具体操作方式请参见应用生命周期管理。
- 在SAE中创建应用后，可以进行弹性自动伸缩，具体操作方式请参见配置弹性伸缩。
- 在SAE中创建应用后，可以通过添加公网 SLB 实现公网访问，添加内网 SLB 实现同 VPC 内所有节点够能通过私网负载均衡访问您的应用，具体操作方式请参见绑定SLB。

问题反馈

如果您在使用 SAE 过程中有任何疑问，欢迎您扫描下面的二维码加入钉钉群进行反馈。

2.2 在 SAE 控制台使用 Jar 包部署微服务应用

应用开发完成后，可以将您的应用部署到 SAE 进行托管。本文介绍如何在 SAE 控制台使用 Jar 包部署微服务应用。

在 SAE 中部署应用的方式如下表所示。

应用举例	部署方式	参考开发文档
原生 Spring Cloud	WAR、JAR、镜像	将 Spring Cloud 应用托管到 SAE
原生 Dubbo	WAR、JAR、镜像	将 Dubbo 应用托管到 SAE
HSF	WAR、JAR、镜像	/
多语言应用	镜像	/

前提条件

- #unique_6/unique_6_Connect_42_section_cu5_k9p_xuf。
- 有环境隔离需求，请#unique_6/unique_6_Connect_42_section_xrz_zr9_py3。

应用 Demo 下载

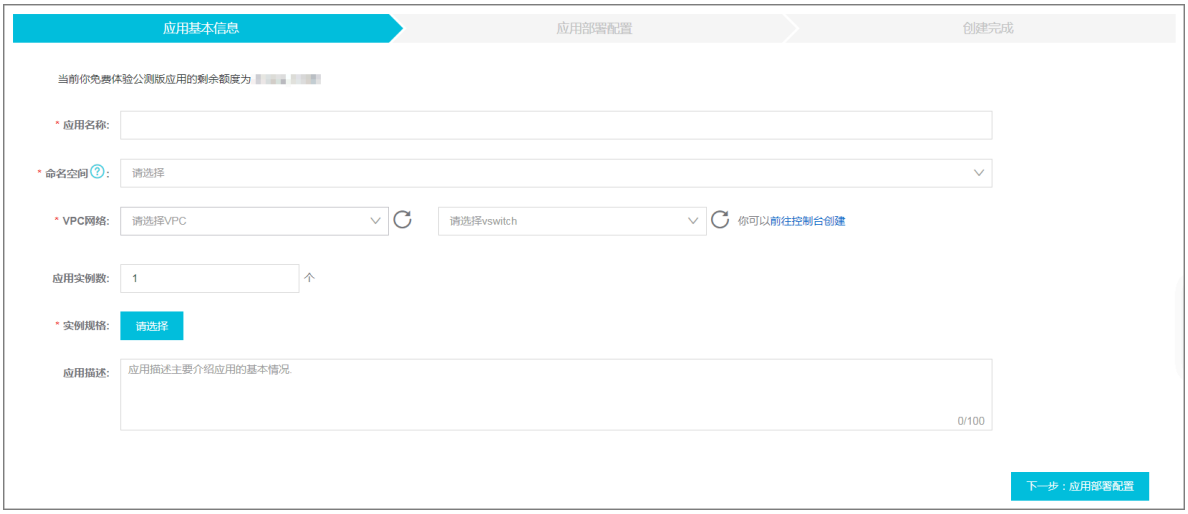
本文以已开发好的 Demo 应用，进行本章内容演示。

- 下载服务提供者应用 Demo： service-provider
- 下载服务消费者应用 Demo： service-consumer

创建服务提供者

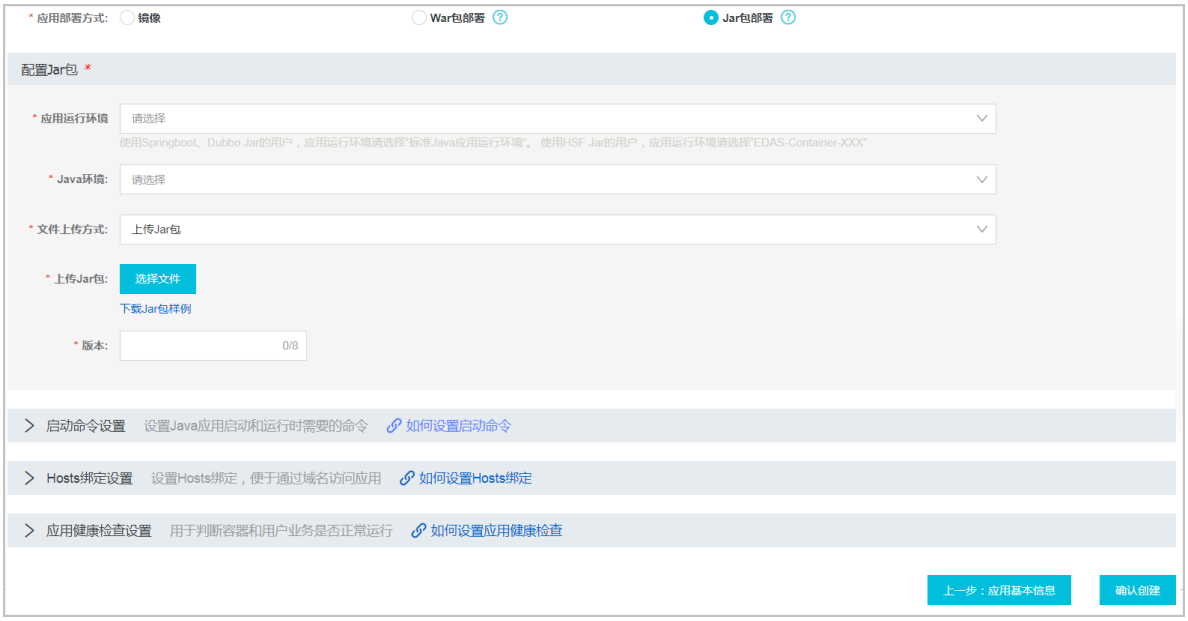
Demo 中的服务提供者蕴含了简易 echo 服务，将自身注册到服务发现中心。

1. 登录 SAE 控制台。
2. 在左侧导航树选择Serverless 应用引擎 > 应用列表，并在 应用列表页面单击右上角 创建应用。
3. 在 应用基本信息页签内，设置应用相关信息，配置完成后单击 下一步：应用部署配置。



- 应用名称：输入应用名称。允许数字，字母，下划线以及中划线组合，仅允许字母开头，最大长度 36 个字符。
- 命名空间：在下拉菜单中选择创建好的命名空间。
- VPC 网络：在下拉菜单中选择VPC 和 vswitch 。
- 应用实例数：选择要创建的实例个数。
- 实例规格：单击请选择，在选择实例规格页面内选择实例的 CPU 和 Memory 规格。
- 应用描述：填写应用的基本情况，输入的描述信息不超过100个字符。

4. 在应用部署配置页面，选择 Jar 包部署，依据页面指示进行配置。完成设置后单击确认创建。



- 应用运行环境：使用 Springboot 或 Dubbo 的用户，应用运行环境请选择*##Java#####*；使用 HSF 的用户，应用运行环境请选择*EDAS-Container-XXX*。
- Java环境：选择 *Open JDK7* 或 *Open JDK8*。
- 文件上传方式：选择*## Jar #*或*Jar ###*。
  - *## Jar #*：下载 service-provider，下载完成后单击选择文件，选择下载完毕的 Jar 包 service-provider 并上传。
  - *Jar ###*：右键单击 service-provider 并选择复制链接地址，将该地址输入在Jar 包地址栏中。

  注意:
  应用部署程序包名仅允许字母、数字，及中划线 “ - ”、下划线 “ _ ” 两个特殊符号。
- 版本：设置版本（如：1.1.0），不建议用时间戳作为版本号。
- 时区设置：选择当前应用所在时区，如*UTC+8*。
- 启动命令设置（可选）：请参见如何设置启动命令配置。
- 环境变量设置（可选）：请参见如何设置环境变量配置。
- Hosts 绑定设置（可选）：请参见如何设置 Hosts 绑定配置。
- 应用健康检查（可选）：请参见如何设置应用健康检查配置。
- 日志收集设置（可选）：请参见如何设置日志收集配置。

5. 在创建完成页面单击应用详情页，查看应用的基本信息和实例部署信息。在实例部署信息页签查看实例的运行状态，如果运行状态显示为绿色的 Running，表示应用部署成功。

创建服务消费者

参见创建服务提供者内容创建服务消费者。

1. 在应用列表页面右上角单击创建应用。
2. 在应用基本信息页面，输入应用相关信息，然后单击下一步：应用部署配置。
   - 应用名称：输入应用名称，此名称须与服务提供者的应用名称不能相同。
   - 命名空间：选择与服务提供者相同的命名空间。
   - VPC 网络：在下拉菜单中选择与服务提供者相同的 VPC 和 vswitch。
   - 应用实例数：选择要创建的实例个数。
   - 实例规格：单击请选择，在选择实例规格页面内选择实例的 CPU 和 Memory 规格。当完成选择后会在应用基本信息页面显示所选择的规格。
   - 应用描述：填写应用的基本情况，输入的描述信息不超过100个字符。
3. 在应用部署配置页面，选择 `Jar 包部署`，并配置 Jar 信息，配置完成后单击确认创建。
   - 应用运行环境：使用 Springboot 或 Dubbo 的用户，应用运行环境请选择*##Java#####*；使用 HSF 的用户，应用运行环境请选择*EDAS-Container-XXX*。
   - Java环境：选择 *Open JDK7* 或 *Open JDK8*。
   - 文件上传方式：选择*## Jar #*或*Jar ###*。
     - *## Jar #*：下载 service-provider，下载完成后单击选择文件，选择下载完毕的 Jar 包 service-provider 并上传。
     - *Jar ###*：右键单击 service-provider 并选择复制链接地址，将该地址输入在Jar 包地址栏中。

     > **注意:**
     > 应用部署程序包名仅允许字母、数字，及中划线 “ - ”、下划线 “ _ ” 两个特殊符号。
   - 版本：设置版本（如：1.1.0），不建议用时间戳作为版本号。
   - 时区设置：选择当前应用所在时区，如*UTC+8*。
   - 启动命令设置（可选）：请参见如何设置启动命令配置。
   - 环境变量设置（可选）：请参见如何设置环境变量配置。
   - Hosts 绑定设置（可选）：请参见如何设置 Hosts 绑定配置。
   - 应用健康检查（可选）：请参见如何设置应用健康检查配置。
   - 日志收集设置（可选）：请参见如何设置日志收集配置。

服务调用验证

1. 在左侧导航栏选择Serverless 应用引擎 > 应用列表。在应用列表中找到所创建的服务提供者和服务消费者应用，单击相应的应用名称进入应用详情页。查看服务列表和实时日志验证应用发布成功并互相调用。
2. 在左侧导航栏单击服务列表，可以分别查看应用发布和消费服务。
   - 在发布的服务页签查看到服务提供者所发布的服务。

   - 在消费的服务页签查服务消费者看到所消费的服务。

3. 为服务消费者应用绑定一个公网SLB，使用绑定后生成的公网IP登录该应用，并开启服务调用。
4. 进入服务消费者应用，在左侧导航树选择应用监控 > 应用总览，查看看调用服务的请求数据。

更多信息

- 使用 SAE 创建应用后，可以进行更新、扩缩容、启停应用监控和删除应用等管理操作，具体操作方式请参见应用生命周期管理。
- 在SAE中创建应用后，可以进行弹性自动伸缩，具体操作方式请参见配置弹性伸缩。
- 在SAE中创建应用后，可以通过添加公网 SLB 实现公网访问，添加内网 SLB 实现同 VPC 内所有节点够能通过私网负载均衡访问您的应用，具体操作方式请参见绑定SLB。

问题反馈

如果您在使用 SAE 过程中有任何疑问，欢迎您扫描下面的二维码加入钉钉群进行反馈。

2.3 在 SAE 控制台使用镜像部署应用

应用开发完成后，可以将您的应用部署到 SAE 进行托管。本文介绍如何在 SAE 控制台使用镜像部署应用。

背景信息

SAE 支持通过镜像部署 SAE 应用，进行应用管理。

SAE 应用的部署方式如下表所示。

应用举例	部署方式	参考开发文档
原生 Spring Cloud	WAR、JAR、镜像	将 Spring Cloud 应用托管到 SAE
原生 Dubbo	WAR、JAR、镜像	将 Dubbo 应用托管到 SAE
HSF	WAR、JAR、镜像	/
多语言应用	镜像	/

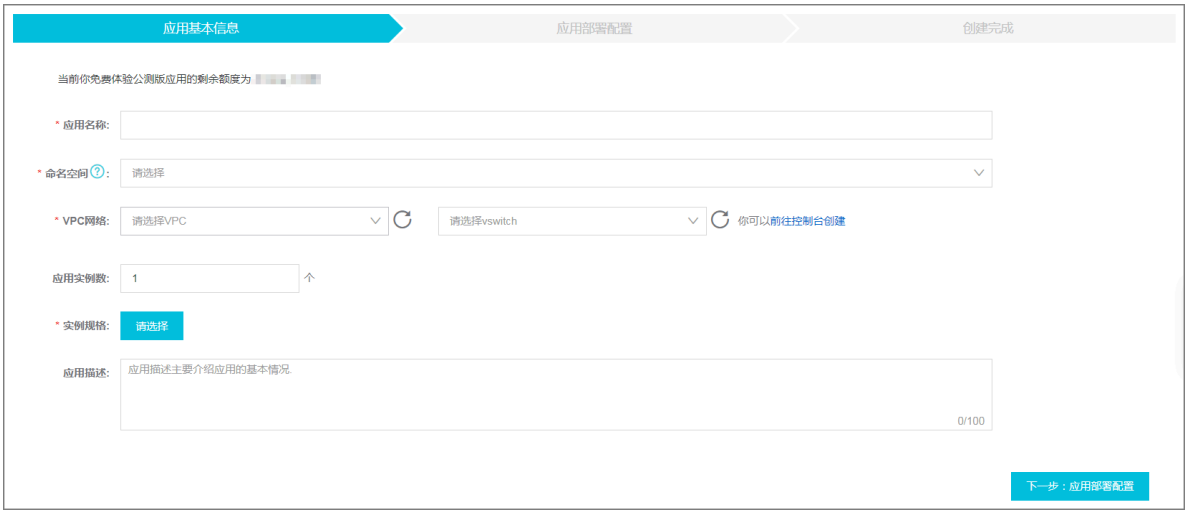
前提条件

1. #unique_6/unique_6_Connect_42_section_xrz_zr9_py3
2. #unique_6/unique_6_Connect_42_section_cu5_k9p_xuf
3. 制作应用镜像

创建 SAE 应用

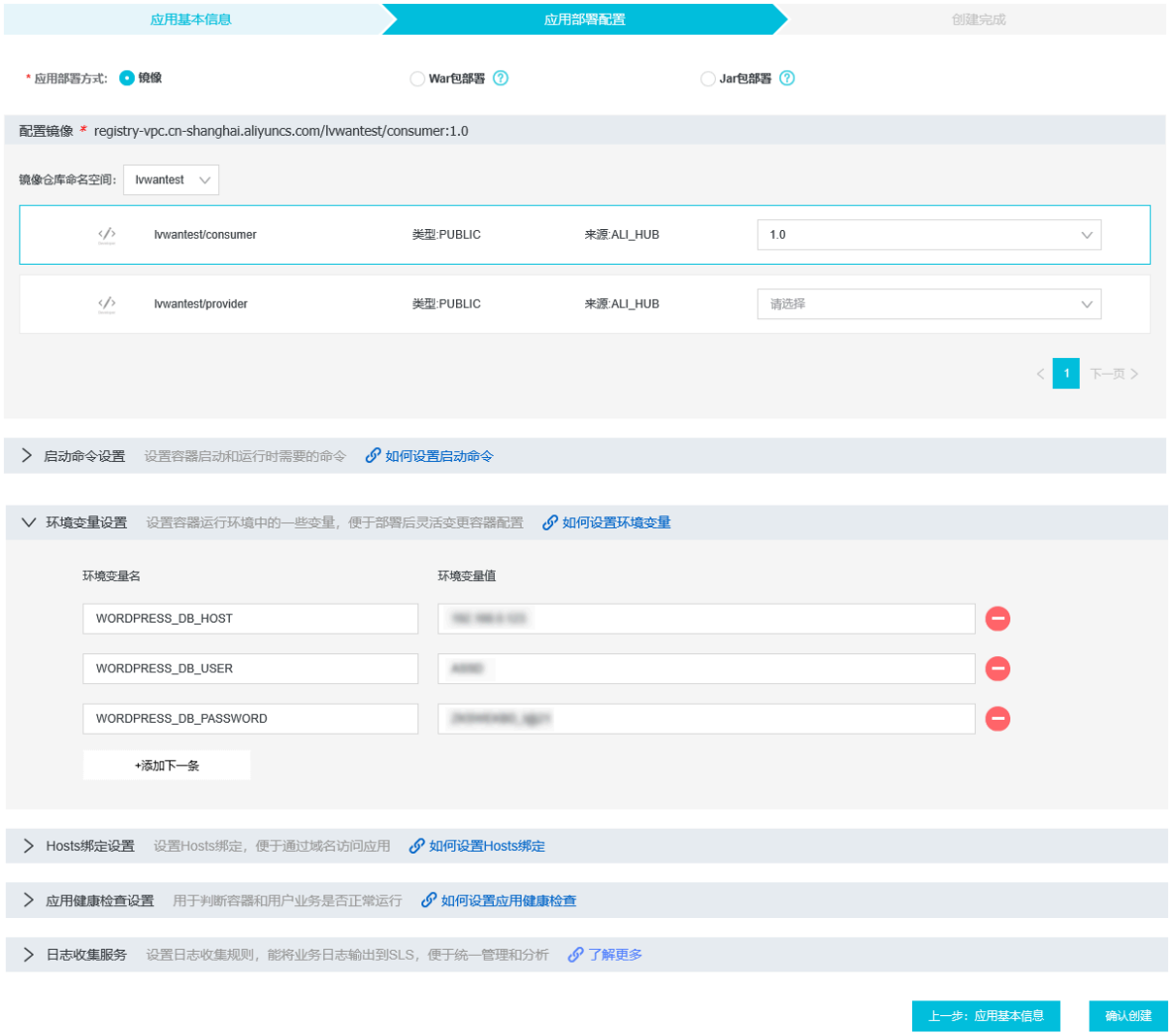
1. 登录 SAE 控制台。
2. 在左侧导航树选择Serverless 应用引擎 > 应用列表，并在 应用列表页面单击右上角 创建应用。

3. 在 应用基本信息页签内，设置应用相关信息，配置完成后单击 下一步：应用部署配置。



- 应用名称：输入应用名称。允许数字，字母，下划线以及中划线组合，仅允许字母开头，最大长度 36 个字符。
- 命名空间：在下拉菜单中选择创建好的命名空间。
- VPC 网络：在下拉菜单中选择VPC 和 vswitch 。
- 应用实例数：选择要创建的实例个数。
- 实例规格：单击请选择，在选择实例规格页面内选择实例的 CPU 和 Memory 规格。
- 应用描述：填写应用的基本情况，输入的描述信息不超过100个字符。

4. 在应用部署配置页面，选择 镜像，依据页面指示进行配置。完成设置后单击确认创建。



- 配置镜像：在镜像列表的下拉框内选择您所创建的镜像，镜像选中后会在配置镜像后面显示您所选择的镜像。
- 启动命令设置（可选）：请参见如何设置启动命令配置。
- 环境变量设置（可选）：请参见如何设置环境变量配置。
- Hosts 绑定设置（可选）：请参见如何设置 Hosts 绑定配置。
- 应用健康检查（可选）：请参见如何设置应用健康检查配置。
- 日志收集设置（可选）：请参见如何设置日志收集配置。

5. 进入应用详情页面，查看应用的基本信息和实例部署信息。

更新应用

部署应用主要用于更新应用，对运行环境变量、镜像配置、启动命令和日志收集等配置项进行修改。

1. 在应用列表中，单击采用镜像部署的 SAE 应用名称，进入应用详情页面。

2. 在页面右上角单击部署应用。
3. 在部署应用页面，重新选择镜像，修改最小存活实例数，并依据现场需要修改配置启动命令、环境变量和健康检查等高级设置选项，配置完成后单击确认。
4. 应用升级过程时，您可以在应用详情页面上方单击查看详情，在变更详情页面查看部署流程和执行日志。部署完成后，如果执行状态为 `Running` ，表示应用更新成功。

结果验证

应用发布后，通过如下两种方式验证应用的发布结果。

- 查看应用实例运行状态。

  在应用详情页中实例部署信息页签查看实例的运行状态。如果执行状态显示为绿色的Running或者 Completed，表示应用发布成功。
- 配置公网负载均衡并访问应用。

  应用发布过程中或发布后，根据实际需要，通过配置负载均衡 SLB 在指定范围内开放应用，以便其它应用访问。

  负载均衡包括如下两类：

  - 私网负载均衡：在应用所在的 VPC 内提供应用的访问入口，保证应用能被同 VPC 内的其它应用访问。
  - 公网负载均衡：为该应用自动购买公网 SLB 服务，保证该应用能被公网中的其它应用访问。

  配置公网 SLB 访问和配置私网 SLB 访问的步骤相同，具体操作请参见#unique_10。SLB 绑定完成后，在浏览器输入由 SLB IP 、端口及应用部署包名组成的访问地址并访问，如 39.106.245.40/80/image，即可进入应用。

更多信息

- 使用 SAE 创建应用后，可以进行更新、扩缩容、启停应用监控和删除应用等管理操作，具体操作方式请参见应用生命周期管理。
- 在SAE中创建应用后，可以进行弹性自动伸缩，具体操作方式请参见配置弹性伸缩。
- 在SAE中创建应用后，可以通过添加公网 SLB 实现公网访问，添加内网 SLB 实现同 VPC 内所有节点够能通过私网负载均衡访问您的应用，具体操作方式请参见绑定SLB。

问题反馈

如果您在使用 SAE 过程中有任何疑问，欢迎您扫描下面的二维码加入钉钉群进行反馈。

2.4 灰度发布或分批发布

灰度发布（又名金丝雀发布），在原有部署版本可用的情况下，同时部署新版本应用作为“金丝雀”（金丝雀对瓦斯极敏感，矿井工人携带金丝雀，以便及时应对危险），测试新版本的性能和表现，在保证整体系统稳定的情况下，能尽早发现和修复问题。

背景信息

分批发布是批次进行应用部署，每次仅对应用的一部分实例进行升级。分批发布过程中如果出现故障，则终止回退，待问题修复后重新发布。

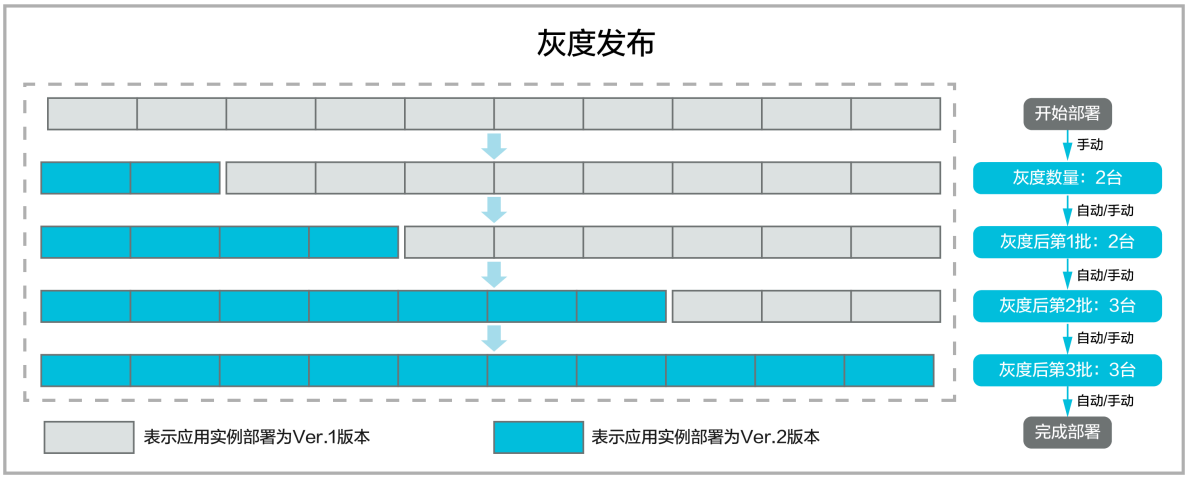
SAE 中当应用进行分批发布时，应用内的实例数平均分配到每个批次进行部署。如果无法平均分配则靠前的批次分配到的示例数少，靠后的批次分配到的示例数少。

当应用进行灰度发布时，为了保证应用稳定性，灰度发布的应用实例数不能超过应用实例总数的50%，剩余的应用实例按照指定的批次分批发布。

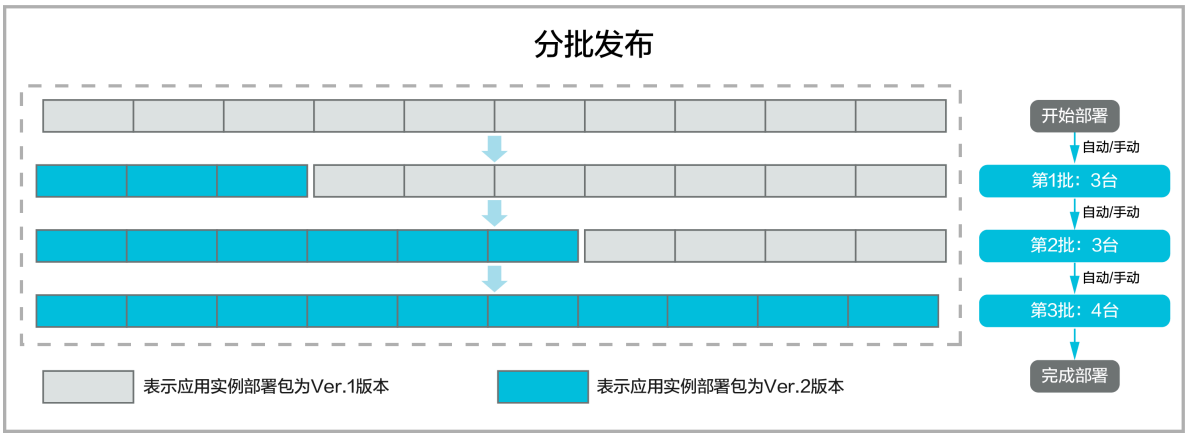
场景示意

某应用包含 10 个应用实例，每个应用实例的部署版本为 Ver.1 版本，现需将每个应用实例升级为Ver.2 版本。

- 灰度发布：假设选择 2 台应用实例进行灰度发布，剩下的 8 个应用实例分 3 批进行分批发布，根据分批发布策略，发布流程如下图所示：



- 分批发布：假设将所有应用实例分 3 批进行部署，根据分批发布策略，该发布流程如下图所示：



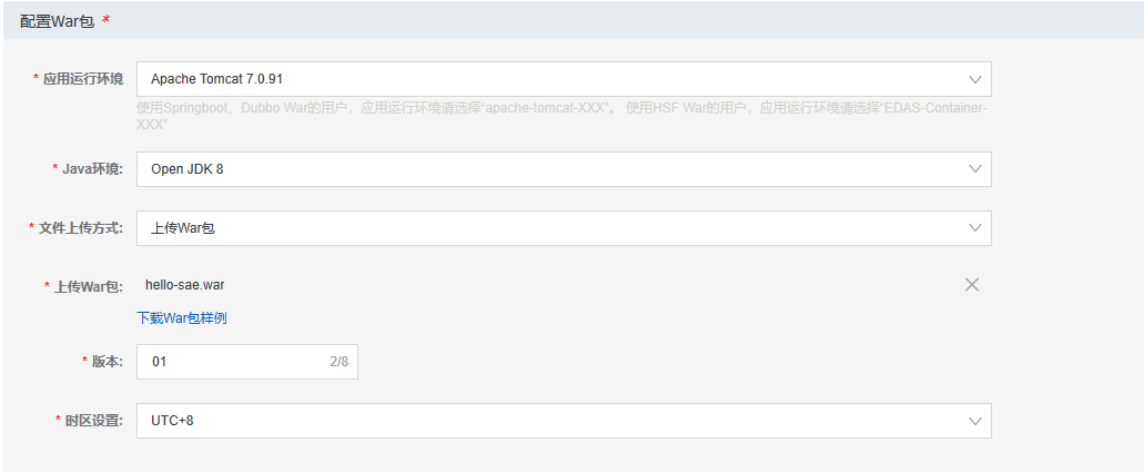
灰度发布

1. 登录 SAE 控制台。
2. 在左侧导航栏中选择应用列表，在应用列表页面中单击需要进行灰度发布的应用名称。
3. 在应用详情页右上角单击部署应用。
4. 配置部署参数。

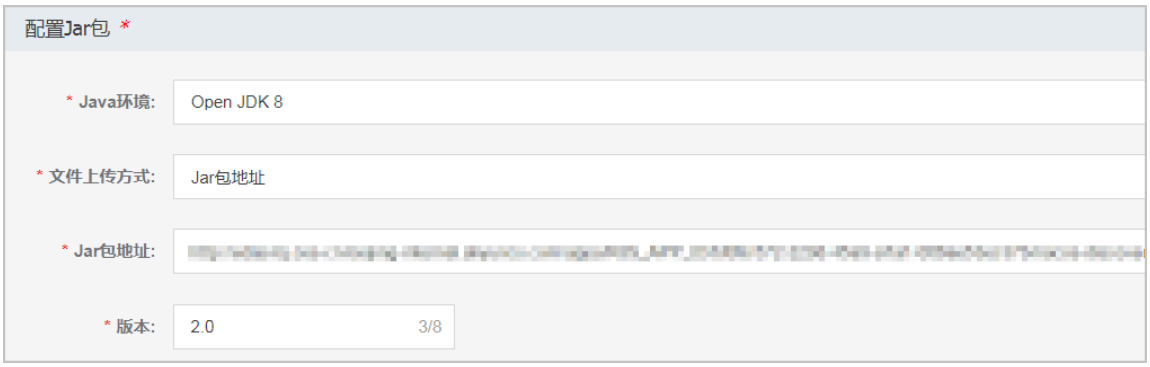
说明:

部署方式由应用首次次部署方式决定，请根据所需的部署方式进行部署参数设置。

- War 包部署：重新上传 War 包或者输入新的部署 War 包的地址，并完成相关环境和参数设置。



- Jar 包部署：重新上传 Jar 包或者输入新部署 Jar 包的地址，并完成相关环境和参数设置。



- 镜像：重新选择镜像。

5. 在发布策略设置区域内配置灰度发布策略。

图 2-1: 灰度发布策略设置

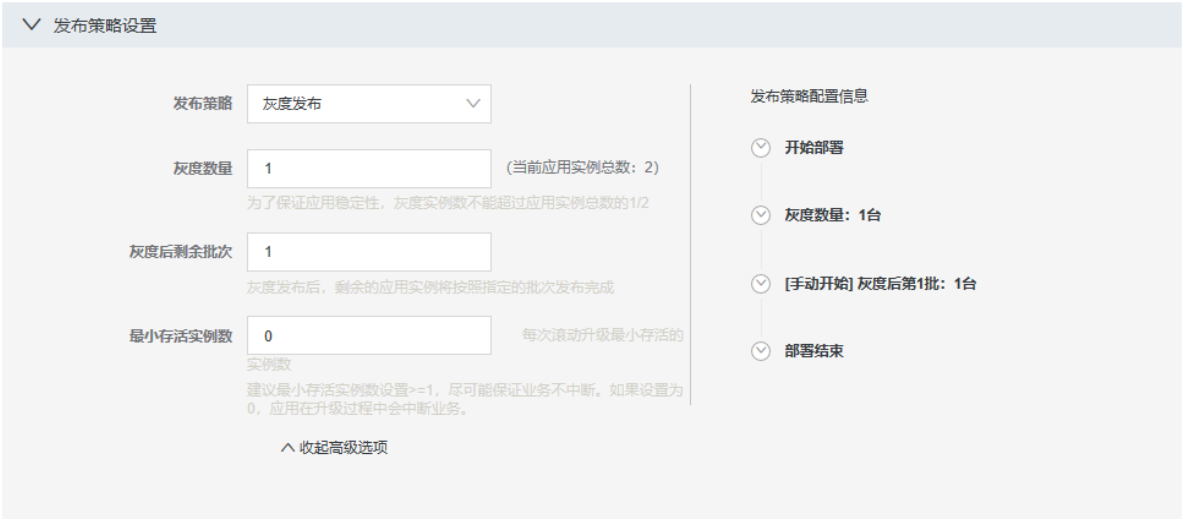


图 2-2: 分批发布策略设置

配置	说明
发布策略	选择灰度发布。
灰度数量	设置需要首先进行灰度发布的应用实例数量。
灰度后剩余批次	灰度发布后，剩余的应用实例按照设定的批次完成发布。
批次内部署间隔	每一批内，如果应用实例数大于 1，应用实例间的部署时间间隔。
最小存活实例数	每次滚动升级最小存活的实例数。

6. 根据实际配置需求，重新配置启动命令、环境变量、Hosts 绑定设置、健康检查和日志收集等高级设置选项。
7. 单击确认完成灰度发布设置。

分批发布

参见灰度发布的过程，设置步骤 5 中的配置策略为####，并依据步骤 5 中的参数配置表进行设置，单击确认进行分批发布。

结果验证

在应用变更记录中查看应用变更详情，查看灰度和分批发布的状态，待所有批次都执行成功则说明应用更新成功。成功后您可以在应用详情页查看实例部署信息，当实例的版本已变为 Ver.2 并且所有实例运行状态均为 Running 时，则说明发布成功。

回滚应用

采用灰度发布或者分批发布方式进行应用升级时，如果应用实例中存在未完成升级的实例，则当前应用升级状态处于进行中。

在实时跟踪升级时，如果首批应用实例升级突发异常停止响应，为了保证业务不受影响，请在变更详情页面单击立即回滚，将已升级的实例回退至升级前版本并将配置还原为升级前原有配置。

在应用变更过程可能出现如遇到部署包不可用、健康检查失败等变更流程异常情况，导致应用升级失败，SAE 将当前应用停止并进行回退。

SAE 应用升级耗时最大为 30 分，超出后系统上报超时异常并暂停变更流程，请在变更详情页面手动终止发布流程并回退。

3 插件部署

3.1 通过 Maven 插件自动部署应用

SAE 应用除通过控制台方式进行应用部署，还可通过 Maven 的 *toolkit-maven-plugin* 插件进行自动化部署。此方式适用于有应用开发经验用户。

自动化部署

通过 *toolkit-maven-plugin* 插件自动化部署应用的流程：添加插件依赖，配置插件，构建部署。

1. 添加插件依赖。

   在 pom.xml 文件中增加如下所示的插件依赖。

```
<build>
    <plugins>
        <plugin>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>toolkit-maven-plugin</artifactId>
            <version>1.0.2</version>
        </plugin>
    </plugins>
</build>
```

2. 配置插件

   配置插件主要包含账号配置，打包配置及部署配置。如果需要更多自定义配置项可参考打包参数和部署参数进行设置。

   a. 账号配置

   在打包工程的根目录下创建文件格式为 `YAML` 的账号配置文件，命名为`toolkit_profile.yaml`并填入如下信息：

```
regionId:        #应用所在区域，如北京为`cn-beijing`，上海为`cn-shanghai`，杭州为`cn-hangzhou`。
accessKeyId:     #访问阿里云资源的AK
accessKeySecret: #访问阿里云资源的SK
```

b. 打包配置

在打包工程的根目录下创建文件格式为 YAML 的打包配置文件。如果打包工程为 Maven 的子模块，则需要在子模块的目录下创建该文件，并命名为toolkit_package.yaml，填入如下信息：

```
apiVersion: V1
kind: AppPackage
spec:
 packageType: #应用部署包类型，支持War、FatJar、Image三种格式。
 packageUrl:  #如果应用部署包类型为War或FatJar，可填入此字段，不填则使用当前maven构建的包进行部署。
 imageUrl:    #如果部署包类型为Image，可填入此字段；Image类型也可以在本地构建Docker镜像进行部署，请参考打包文件参数章节设置相关参数。
```

c. 部署配置

在打包工程的根目录下创建文件格式为 YAML 的部署文件，命名为toolkit_deploy.yaml，并填入如下信息：

```
apiVersion: V1
kind: AppDeployment
spec:
type: serverless
target:
 appId:         #部署应用的ID，如果配置了appId则无需配置namespaceId和appName
 namespaceId:   #所属区域，如不清楚appId，可使用此所属区域及应用名称进行部署
 appName:       #应用名称，如不清楚appId，可使用此应用名称及命名空间进行部署
```

3. 构建部署

进入pom.xml所在的目录（如果部署 Maven 子模块，则进入子模块pom.xml所在的目录），执行如下命令：

```
mvn clean package toolkit:deploy -Dtoolkit_profile=toolkit_profile.yaml -Dtoolkit_package=toolkit_package.yaml -Dtoolkit_deploy=toolkit_deploy.yaml
```

命令参数含义为：

- `toolkit:deploy`：在打包完成后进行应用部署。
- `-Dtoolkit_profile`：指定账号配置文件。如果账号文件跟 `pom.xml` 在同一个目录下，且名字为`.toolkit_profile.yaml`（注意：文件名最前面有个小数点），可不填此参数，插件会自动获取。
- `-Dtoolkit_package`: 指定打包文件。如果打包文件跟 `pom.xml` 在同一个目录下，且名字为`.toolkit_package.yaml`（注意：文件名最前面有个小数点），可不填此参数，插件会自动获取。
- `-Dtoolkit_deploy`: 指定部署文件。如果部署文件跟 `pom.xml` 在同一个目录下，且名字为`.toolkit_deploy.yaml`（注意：文件名最前面有个小数点），可不填此参数，插件会自动获取取。

执行该打包命令后，系统显示如下结果，当回显信息中显示“BUDILD SUCCESS”表示部署成功。

```
[INFO] Sending deploy request to EDAS...
[INFO] Deploy request sent, changeOrderId is: 71548877-5bb3-4503-88e5-04761c0ac688
[INFO] Begin to trace change order: 71548877-5bb3-4503-88e5-04761c0ac688
[INFO] PipelineName:Batch: 1, PipelineId:9d7b8038-534c-4891-8f70-e63db6606297
[INFO] StageName:Workflow Start, StageId:cda21377-260d-43f0-9c82-98b55c3fd1ec
[INFO] ServiceStageName:Workflow Start, ServiceStageId:cda21377-260d-43f0-9c82-98b55c3fd1ec
[INFO] StageName:SLB Offline, StageId:1060b156-38b3-47aa-990d-5b5925256a57
[INFO] ServiceStageName:SLB Offline, ServiceStageId:1060b156-38b3-47aa-990d-5b5925256a57
[INFO] StageName:Deploy, StageId:035c7222-b84a-4c3a-bec2-3aeb3a5ac90e@co_virtual_stage
[INFO] InstanceName:nahai-20180920, InstanceIp:47.93.157.199(Public)<br>10.30.137.182(Inner)
[INFO] InstanceStageName:RPC Offline, InstanceStageId:3a0fbf0b-1dda-4368-8b67-b2e8bf7c7de7
[INFO] Waiting...
[INFO] Waiting...
[INFO] Waiting...
[INFO] Waiting...
[INFO] Waiting...
[INFO] Waiting...
[INFO] InstanceStageName:Stop Application, InstanceStageId:f8df6de7-adea-47c0-a2cd-6a18b49ebb2f
[INFO] InstanceStageName:Environment initial, InstanceStageId:033ca419-7d0e-4996-ba91-0b4e67877548
[INFO] InstanceStageName:Start Application, InstanceStageId:634e45bc-a411-429d-a549-b693794a41ac
[INFO] Waiting...
[INFO] Waiting...
[INFO] InstanceStageName:Health Check, InstanceStageId:77f179ee-f8ac-40ad-85a5-86cad41b1e4a
[INFO] Waiting...
[INFO] Waiting...
[INFO] Waiting...
[INFO] Waiting...
[INFO] StageName:SLB Online, StageId:035c7222-b84a-4c3a-bec2-3aeb3a5ac90e
[INFO] ServiceStageName:SLB Online, ServiceStageId:035c7222-b84a-4c3a-bec2-3aeb3a5ac90e
[INFO] StageName:Workflow Complete, StageId:510d5110-d430-4508-93c4-864dc506b9ae
[INFO] ServiceStageName:Workflow Complete, ServiceStageId:510d5110-d430-4508-93c4-864dc506b9ae
[INFO] Deploy application successfully!
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

更多配置项

1. 打包参数

打包文件支持的参数如下所示：

```
apiVersion: V1
kind: AppPackage
spec:
  packageType:  #应用部署包类型，支持War、FatJar、Image。
  imageUrl:     #镜像地址，Image包类型应用可填入。
  packageUrl:   #包地址，War、FatJar类型应用可填入。
  build:
    docker:
      dockerfile:         #Docker镜像构建文件。如您希望在本地构建镜像部
署，需填入此字段。
      imageRepoAddress:   #阿里云镜像仓库地址。如您希望在本地构建镜像部
署，需填入此字段。
      imageTag:           #镜像Tag。如您希望在本地构建镜像部署，需填入此字
段。
      imageRepoUser:      #阿里云镜像仓库用户名。如您希望在本地构建镜像部
署，需填入此字段。
      imageRepoPassword: #阿里云镜像仓库密码。如您希望在本地构建镜像部
署，需填入此字段。
    oss:
      bucket:       #目标存储桶名称。如您希望使用自定义的oss仓库存储部署
包，需填入此字段。
      key:          #oss自定义路径。如您希望使用自定义的oss仓库存储部署
包，需填入此字段。
      accessKeyId: #oss账号。如您希望使用自定义的oss仓库存储包，需填入此字
段。
      accessKeySecret: #oss密码。如您希望使用自定义的oss仓库存储包，可填入
此字段。
```

2. 部署参数

部署文件支持的参数如下所示：

```
apiVersion: V1
kind: AppDeployment
spec:
  type: serverless
  target:
    appName:     #应用名称
    namespaceId: #应用所在命名空间
    appId:       #应用ID。插件会使用此应用进行部署，如未填入则使用namespaceI
d和appName来查找应用进行部署。
  version: #部署版本号，默认使用日时分秒格式
  jdk:     #部署的包依赖的 JDK 版本，JDK 支持版本为 Open JDK 7 和 Open
JDK 8。镜像不支持。
  webContainer:  #部署的包依赖的 Tomcat 版本，WebContainer 支持 apache-
tomcat-7.0.91。镜像不支持。
  batchWaitTime: #分批等待时间。
  command:       #镜像启动命令。该命令必须为容器内存在的可执行的对象。例如：
sleep。设置该命令将导致镜像原本的启动命令失效。
  commandArgs: #镜像启动命令参数。上述启动命令所需参数。
    - 1d
  envs:    #容器环境变量参数
    - name: envtmp0
      value: '0'
```

```
    - name: envtmp1
      value: '1'
  customHostAlias:  #容器内自定义host映射
    - hostName: 'samplehost1'
      ip: '127.0.0.1'
    - hostName: 'samplehost2'
      ip: '127.0.0.1'
  jarStartOptions:  #Jar包启动应用选项
  jarStartArgs:     #Jar包启动应用参数
  liveness:   #容器健康检查，健康检查失败的容器将被杀死并恢复。
    exec:
      command:
        - sleep
        - 1s
    initialDelaySeconds: 5
    timeoutSeconds: 11
  readiness:  #应用启动状态检查，多次健康检查失败的容器将被杀死并重启。不通过健康检查的容器将不会有 SLB 流量进入。
    exec:
      command:
        - sleep
        - 1s
    initialDelaySeconds: 5
    timeoutSeconds: 11
  minReadyInstances: 1  #最小存活实例数。在滚动升级过程中或者滚动升级失败，可用实例数都将尽可能不小于该值，为 0 则不限制。弹性扩缩容的范围不应该小于该值，否则将触发异常。
```

典型场景示例

典型部署场景及相关配置示例。

- 场景一：本地构建War（或FatJar）包进行部署

  假设您在北京环境有 WAR（或 FatJar）类型的 SAE 应用，期望本地构建 WAR（或 FatJar）进行部署。打包配置和部署配置如下所示。

  - 打包文件：

    ```
    apiVersion: V1
    kind: AppPackage
    spec:
      packageType: War
    ```

  - 部署文件：

    ```
    apiVersion: V1
    kind: AppDeployment
    spec:
      type: serverless
      target:
        appId:  #应用ID。插件会使用此应用进行部署，如未填入则使用namespaceId和appName来查找应用进行部署。
        namespaceId:  #【可选】命名空间，如不清楚appId，可使用此命名空间及应用名称进行部署
        appName:      #【可选】应用名称，如不清楚appId，可使用此命名空间及应用名称进行部署
    ```

```
```

- 场景二：使用已有镜像地址部署镜像类型应用

  假设您在北京环境有一个镜像类型应用，期望使用已有的镜像（registry.cn-beijing.aliyuncs.com/test/gateway:latest ）部署应用。打包配置和部署配置如下所示。

  - 打包文件：

```
apiVersion: V1
kind: AppPackage
spec:
  packageType: Image
  imageUrl: registry.cn-beijing.aliyuncs.com/test/gateway:latest
```

  - 部署文件：

```
apiVersion: V1
kind: AppDeployment
spec:
  type: serverless
  target:
    appId:  #应用ID。插件会使用此应用进行部署，如未填入则使用namespaceId和appName来查找应用进行部署。
    namespaceId:  #【可选】命名空间，如不清楚appId，可使用此命名空间及应用名称进行部署
    appName:      #【可选】应用名称，如不清楚appId，可使用此命名空间及应用名称进行部署
```

- 场景三：本地构建镜像上传至仓库并部署应用

  假设您在北京环境有镜像类型应用，期望在本地编译并构建为镜像，并上传到阿里云镜像仓库进行部署，打包配置和部署配置如下所示。

  - 打包文件：

```
apiVersion: V1
kind: AppPackage
spec:
  packageType: Image
  build:
    docker:
       dockerfile: Dockerfile #指定Dockerfile
       imageRepoAddress:      #镜像仓库地址
       imageTag:              #镜像Tag
       imageRepoUser:         #镜像仓库用户名
       imageRepoPassword:     #镜像仓库密码
```

  - 部署文件：

```
apiVersion: V1
kind: AppDeployment
spec:
  type: serverless
  target:
```

```
    appId:   #应用ID。插件会使用此应用进行部署，如未填入则使用namespaceId和
appName来查找应用进行部署。
    namespaceId:   #【可选】命名空间，如不清楚appId，可使用此命名空间及应用
名称进行部署
    appName:       #【可选】应用名称，如不清楚appId，可使用此命名空间及应用
名称进行部署
```

3.2 通过 IntelliJ IDEA 插件部署应用

SAE 应用除通过控制台方式进行应用部署，还可通过 Alibaba Cloud Toolkit for IntelliJ IDEA 插件部署应用。此方式适用于有应用开发经验的用户。

背景信息

Alibaba Cloud Toolkit for IntelliJ IDEA（本文简称 Cloud Toolkit）为阿里巴巴提供的免费 IDE 插件。注册或使用已有阿里云账号免费下载 Cloud Toolkit。下载完成后，将其安装在 IntelliJ IDEA 中。

在本地完成应用程序的开发、调试及测试后，通过本插件将应用程序快速部署到阿里云。目前仅支持将应用部署到 ECS、EDAS、SAE 或容器服务 Kubernetes。

前提条件

- 下载并安装 JDK 1.8 或更高版本。
- 下载并安装 IntelliJ IDEA (2018.3 或更高版本)。

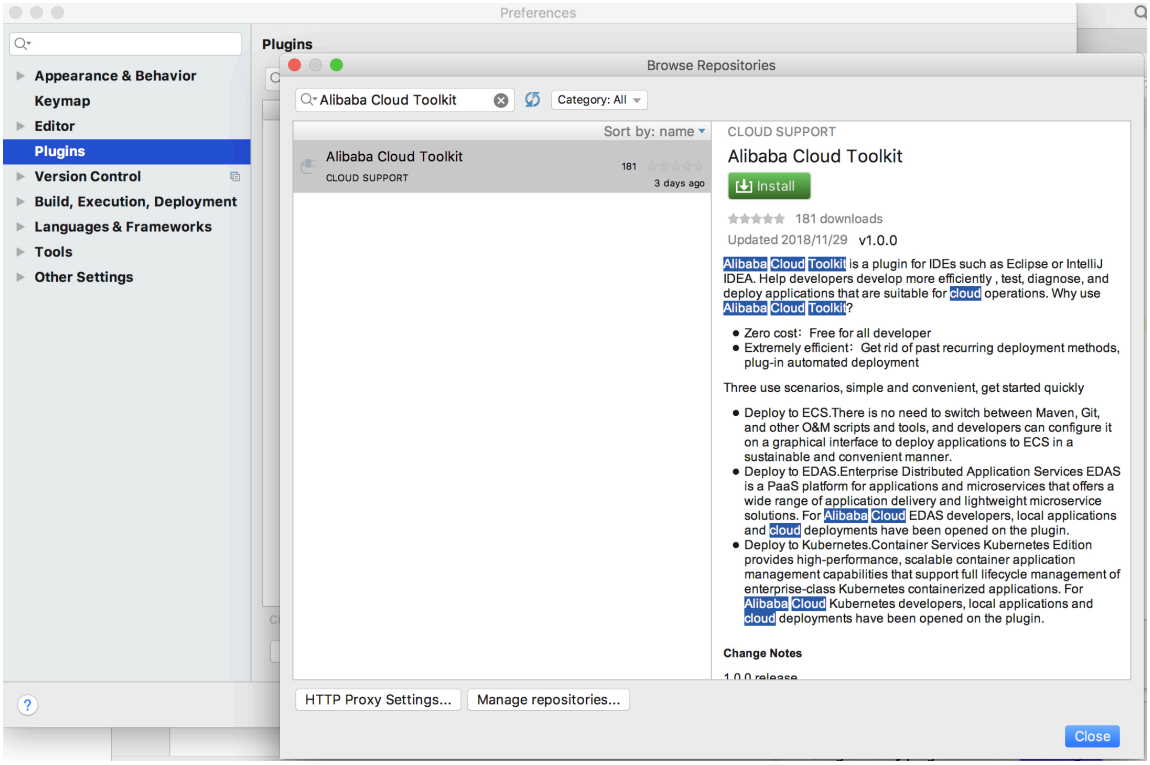
> **说明:**
> 由于 JetBrains 插件官方服务器设立在海外，如果因访问缓慢导致无法下载安装，请加入文末交流群，从 Cloud Toolkit 产品运营部获取离线安装包。

安装 Cloud Toolkit

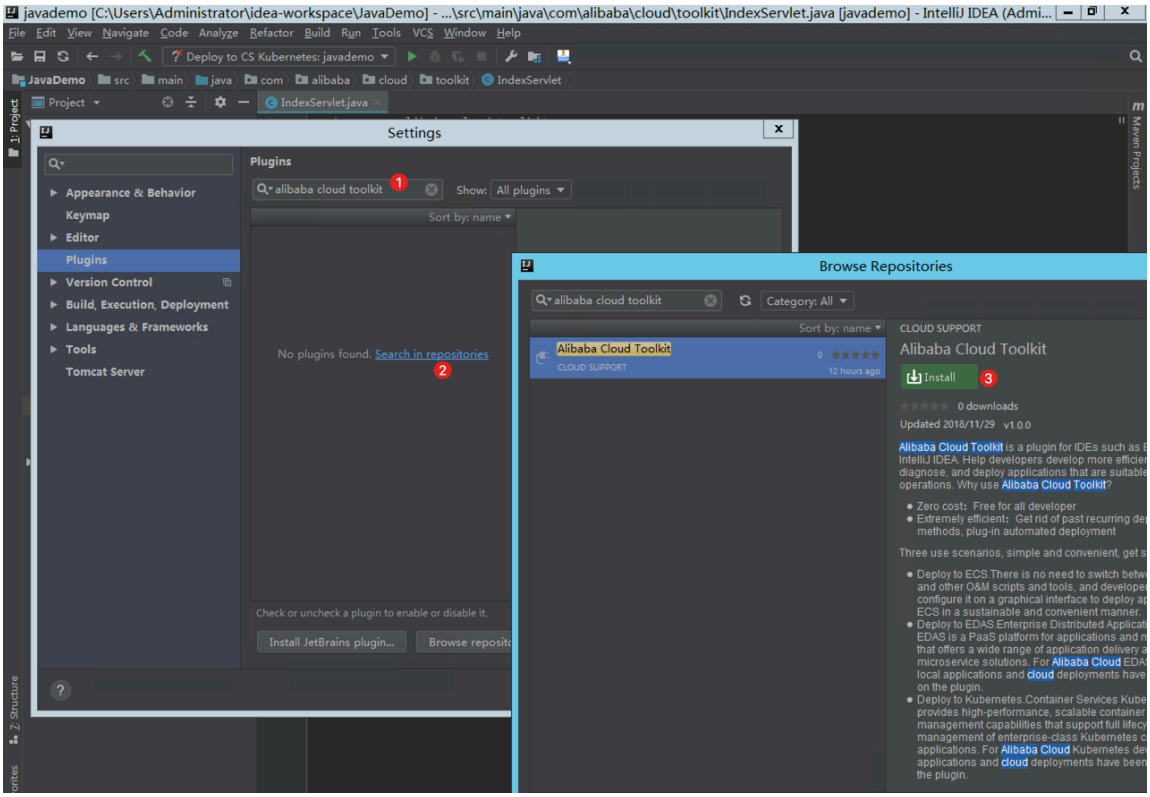
1. 启动 IntelliJ IDEA。

2. 在 IntelliJ IDEA 中安装插件。

   - Mac 系统：进入 Preference 配置页面，选择左边的 `Plugins`，在右边的搜索框里输入 `Alibaba Cloud Toolkit`，并单击Install 安装。



   - Windows 系统：进入 Plugins 选项，搜索 `Alibaba Cloud Toolkit`，并单击Install安装。

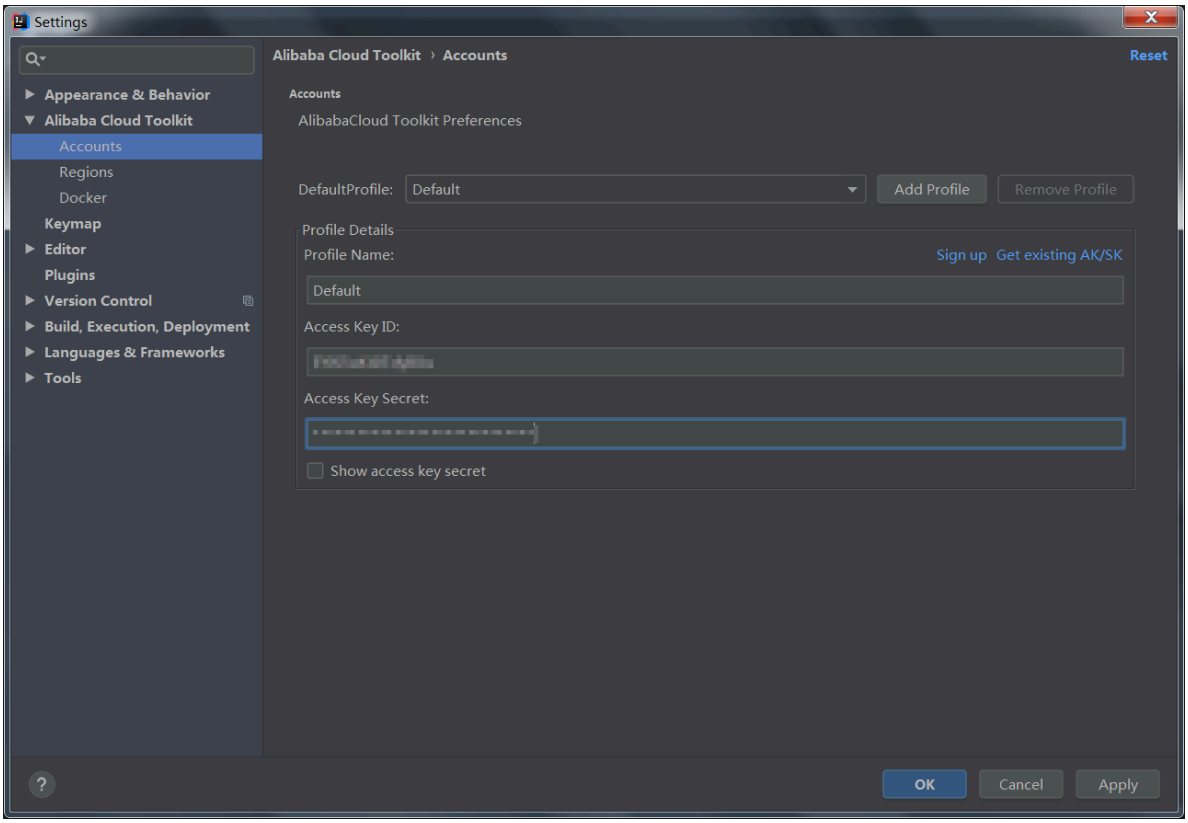


3. 插件安装成功后，重启 IntelliJ IDEA，在工具栏显示 Cloud Toolkit 图标（）。

配置 Cloud Toolkit 账号

安装 Cloud Toolkit 完成后，需要使用 Access Key ID 和 Access Key Secret 配置 Cloud Toolkit 账号。

1. 启动 IntelliJ IDEA。
2. 单击 Cloud Toolkit 图标，在下拉列表中单击 Preference…，在设置页面左侧导航栏选择 Alibaba Cloud Toolkit > Accounts 。
3. 在 Accounts 界面中设置Access Key ID 和 Access Key Secret，并单击 OK。



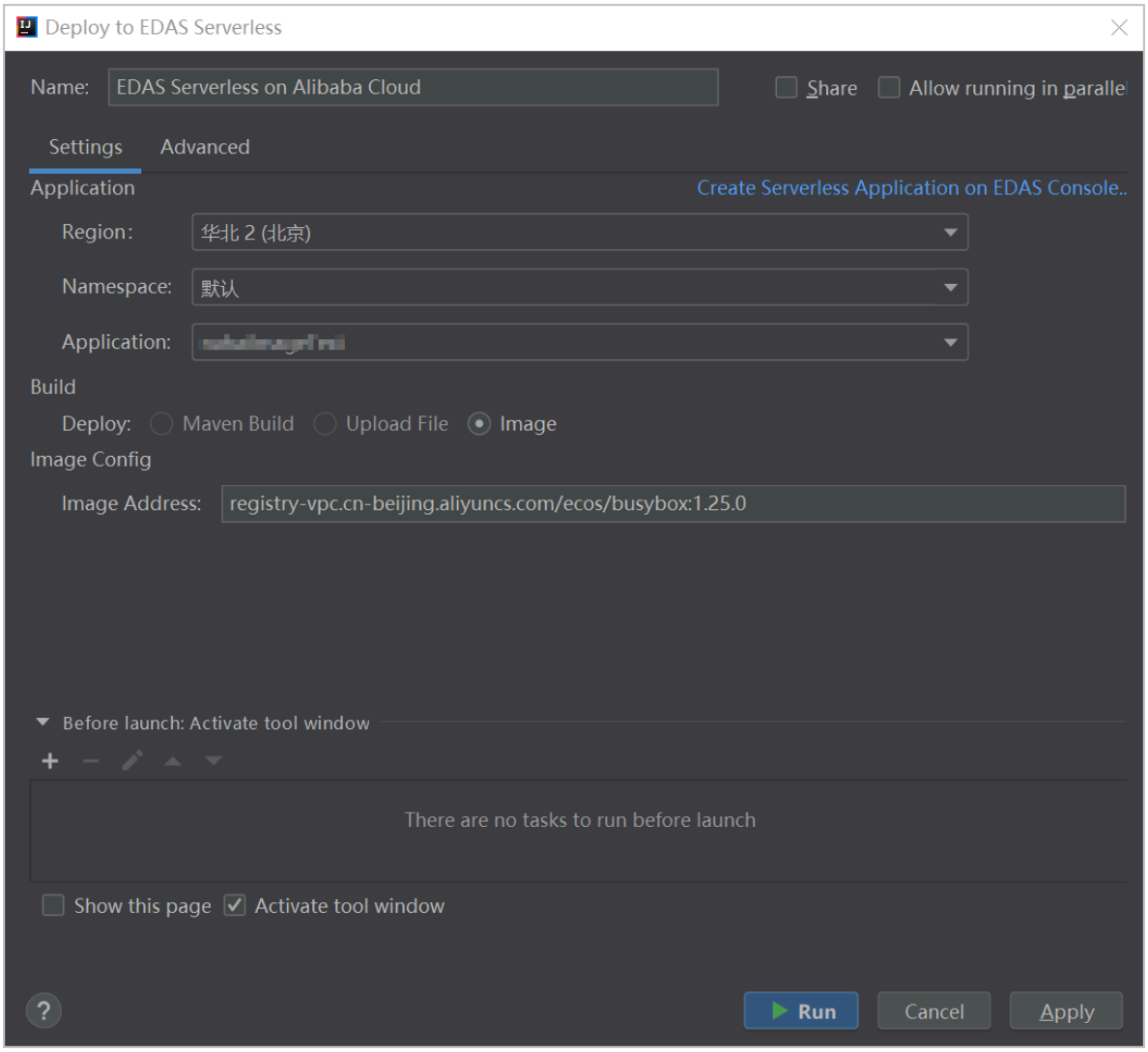
- 已有阿里云账号，在 Accounts 界面中单击 Get existing AK/SK，进入并登录阿里云登录页面，系统自动跳转至安全信息管理页面，获取 Access Key ID 和 Access Key Secret。
- 没有阿里云账号，在 Accounts 界面中单击 Sign up，进入阿里云账号注册页面并完成注册。注册完成后并获取 Access Key ID 和 Access Key Secret。

部署应用到 SAE

Cloud Toolkit 插件目前仅支持将应用以 WAR包、JAR包或Image部署到 SAE。

1. 在 IntelliJ IDEA 上单击 Cloud Toolkit 图标，并在下拉列表中选择 Deploy to EDAS Serverless。

2. 在 Deploy to EDAS Serverless 运行配置页面，配置应用部署参数。配置完成后单击 Apply 保存设置。



a. 在配置页面中根据需求选择应用的 `Region`、`Namespace`、`Application` 。

   `Region`：应用所在地；`Namespace`：应用所在命名空间；`Application`：应用名称。

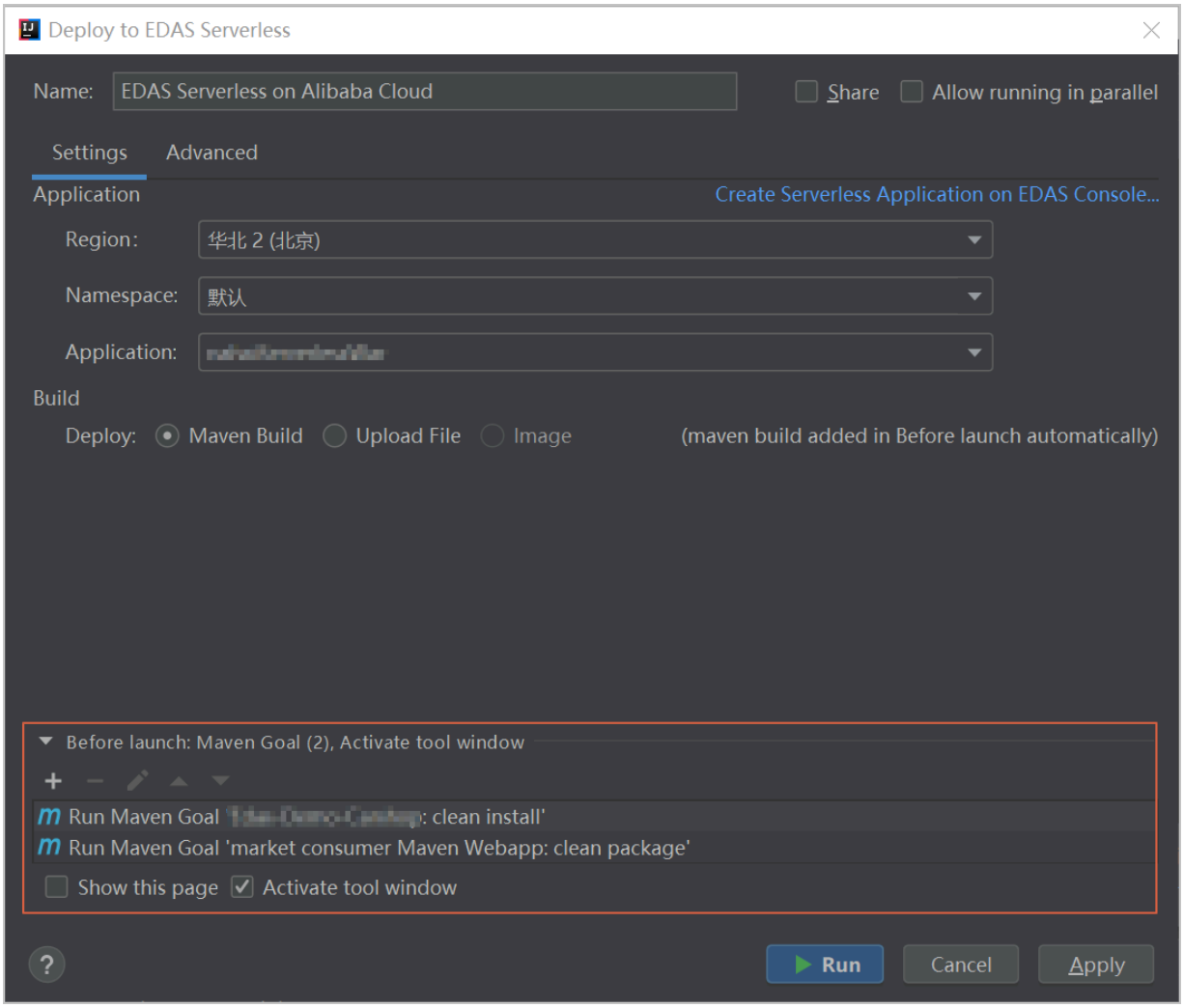
b. 设置构建方式。

   - Maven Build：选择 Maven Build 方式构建应用，系统默认添加 Maven 任务构建部署包。如果需要部署子模块，具体操作请参见部署多模块工程增加相应的 Maven 任务。
   - Upload File：选择 Upload File 方式构建应用，上传待部署的 WAR 包或者 JAR 包并进行部署。
   - Image：选择 Image 方式构建应用，注入镜像地址并进行部署。

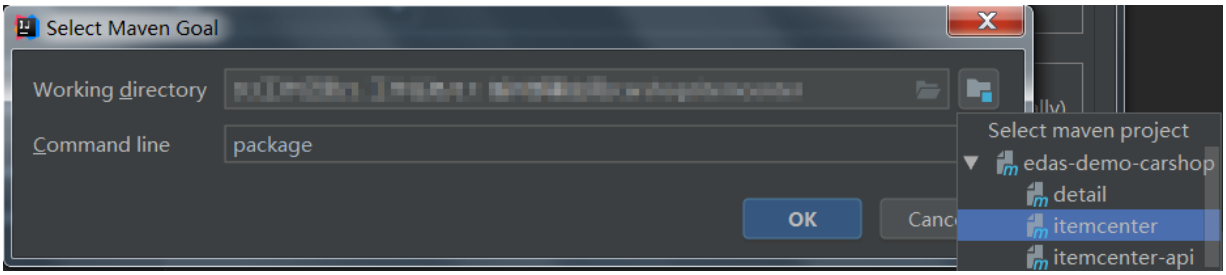
3. 单击 Run，IntelliJ IDEA 的 Console 区域打印部署应用运行日志。您可以根据日志信息检查部署结果。

管理 Maven 构建任务

在 Deploy to EDAS Serverless 配置页面的 Before launch 区域，可以对进行 Maven 构建任务添加、删除、修改和移动等管理操作。



在添加 Maven 构建任务编辑框中，单击右侧的文件夹按钮并选择当前工程所有可用模块，在 `Command line` 中输入构建命令。



部署多模块工程

Maven 工程为多模块开发模式，各个模块独立开发，模块之前存在调用关系，这样的项目工程称为多模块工程。

需要部署多模块 Maven 工程的子模块，在 EDAS Serverless Deployment 配置页面的 Before launch 中，将待部署的子模块任务设置为最后执行，如何设置具体操作请参见管理 Maven 构建任务 。

例如：当前 CarShop 工程存在如下子模块：

carshop

- itemcenter-api
- itemcenter
- detail

其中，itemcenter 和 detail 为子模块，且都依赖于 itemcenter-api 模块。现在需要部署 itemcenter 模块，则在配置页面中 `Before launch` 中增加如下两个 Maven 任务。

1. 在父工程 carshop 中增加执行 `mvn clean install` 的 Maven 任务。
2. 在子模块 itemcenter 中增加执行 `mvn clean package`的 Maven 任务。

注意:

确保子模块的 Maven 任务为 `Before launch` 最后执行的构建任务。

问题反馈

如果您在使用 SAE 过程中有任何疑问，欢迎您扫描下面的二维码加入钉钉群进行反馈。

3.3 通过 Eclipse 插件一键部署应用

SAE 应用除通过控制台方式进行应用部署，还可以通过 Alibaba Cloud Toolkit for Eclipse 插件部署应用。此方式适用于有应用开发经验的用户。

背景信息

Alibaba Cloud Toolkit for Eclipse（本文简称 Cloud Toolkit）为阿里巴巴提供的免费 IDE 插件。注册或使用已有阿里云账号免费下载 Cloud Toolkit。下载完成后，将其安装在 IntelliJ IDEA 中。

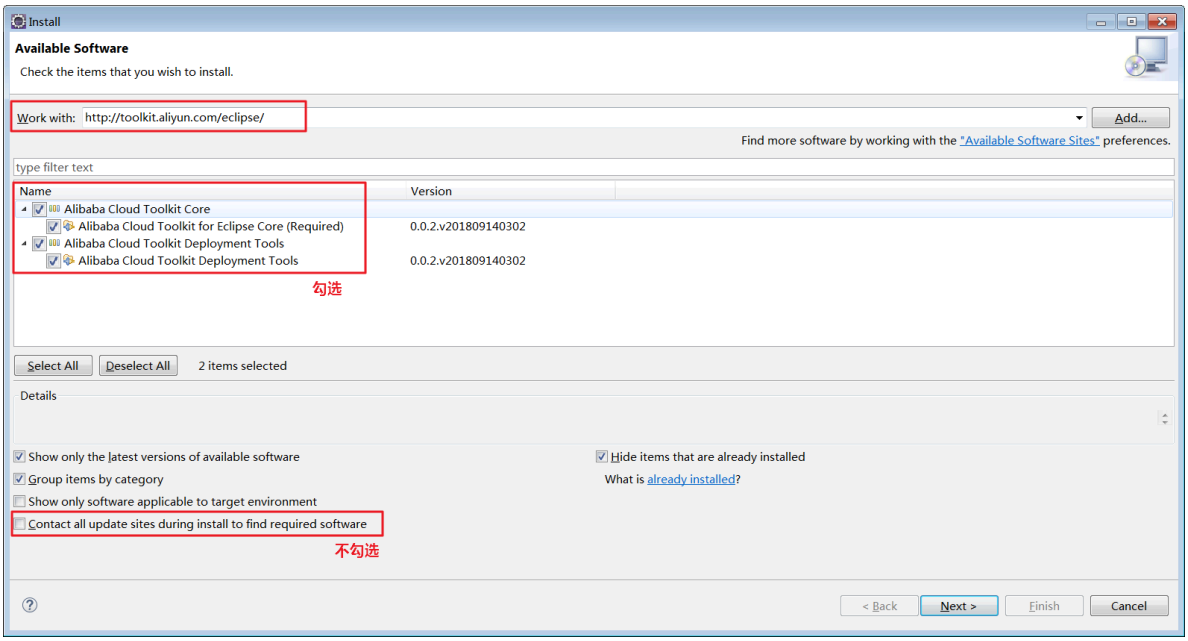
在本地完成应用程序的开发、调试及测试后，通过本插件将应用程序快速部署到阿里云。目前仅支持将应用部署到 ECS、EDAS、SAE 或容器服务 Kubernetes。

前提条件

- 下载并安装 JDK 1.8 或更高版本
- 下载并安装适用于 Java EE 开发的 Eclipse IDE、4.5.0（代号：Mars）或更高版本

安装 Cloud Toolkit

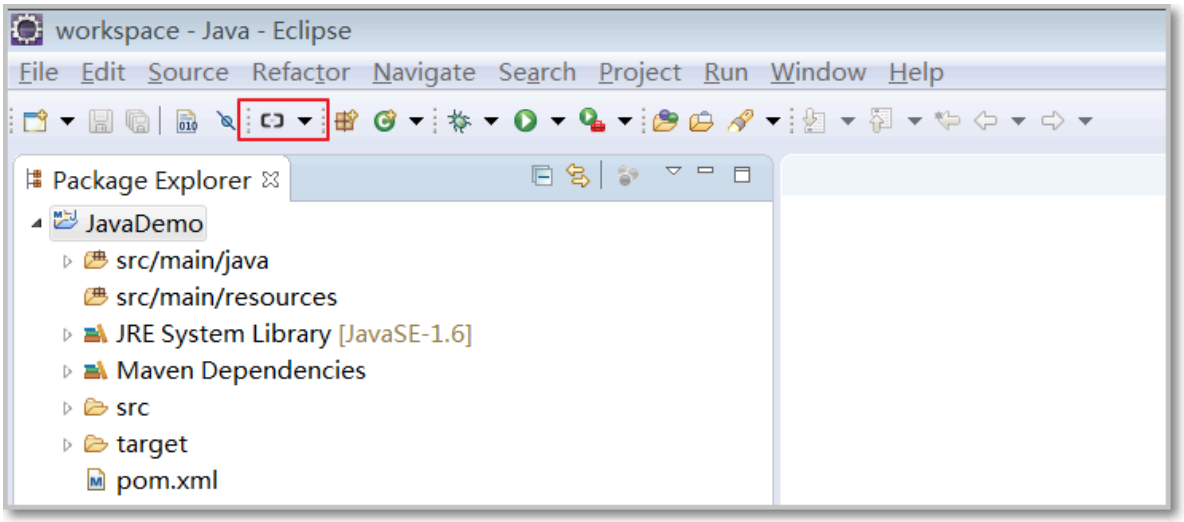
1. 启动 Eclipse。
2. 在菜单栏中选择Help > Install New Software。
3. 在 Available Software 对话框的 Work with 文本框中，输入 `Cloud Toolkit for Eclipse` 的 URL。
4. 在下图所示的列表区域中勾选需要的组件 Alibaba Cloud Toolkit Core 和 Alibaba Cloud Toolkit Deployment Tools，并在下方 Details 区域中去勾选 Connect all update sites during install to find required software，取消勾选后单击Next。



5. 按照 Eclipse 安装页面的提示，完成后续安装步骤。

> **注意:**
> 安装过程中提示没有数字签名对话框，请选择 Install anyway。

6. Cloud Toolkit 插件安装完成后，重启 Eclipse。重启后在工具栏显示 Alibaba Cloud Toolkit 图标。

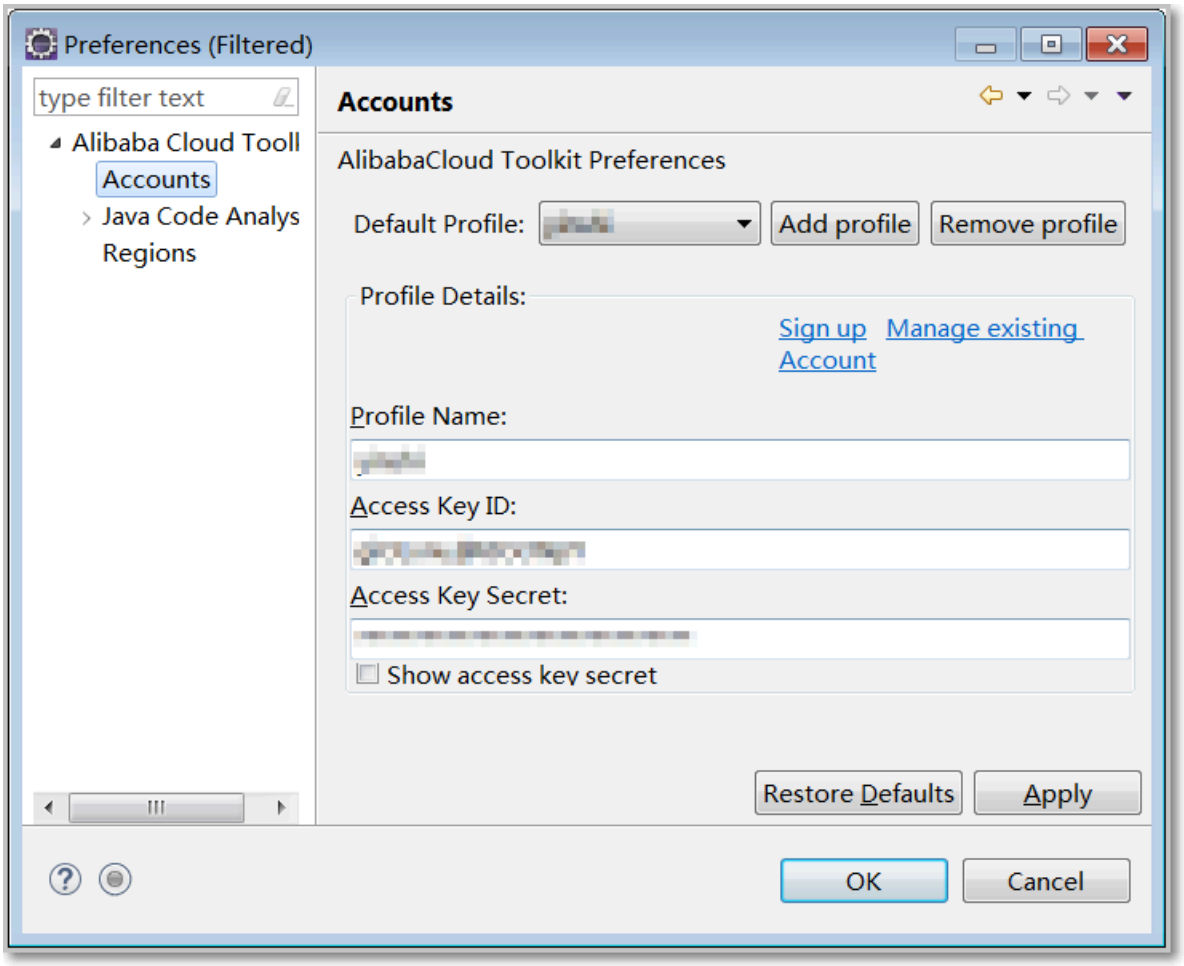


配置 Cloud Toolkit 账号

安装 Cloud Toolkit 完成后，需要使用 Access Key ID 和 Access Key Secret 配置 Cloud Toolkit 账号。

1. 启动 Eclipse。
2. 在工具栏 Cloud Toolkit 图标右侧下拉菜单中选择 Alibaba Cloud Preference… > Alibaba Cloud Tooll > Accounts。

3. 在 Accounts 界面中设置 Access Key ID 和 Access Key Secret并单击 OK。



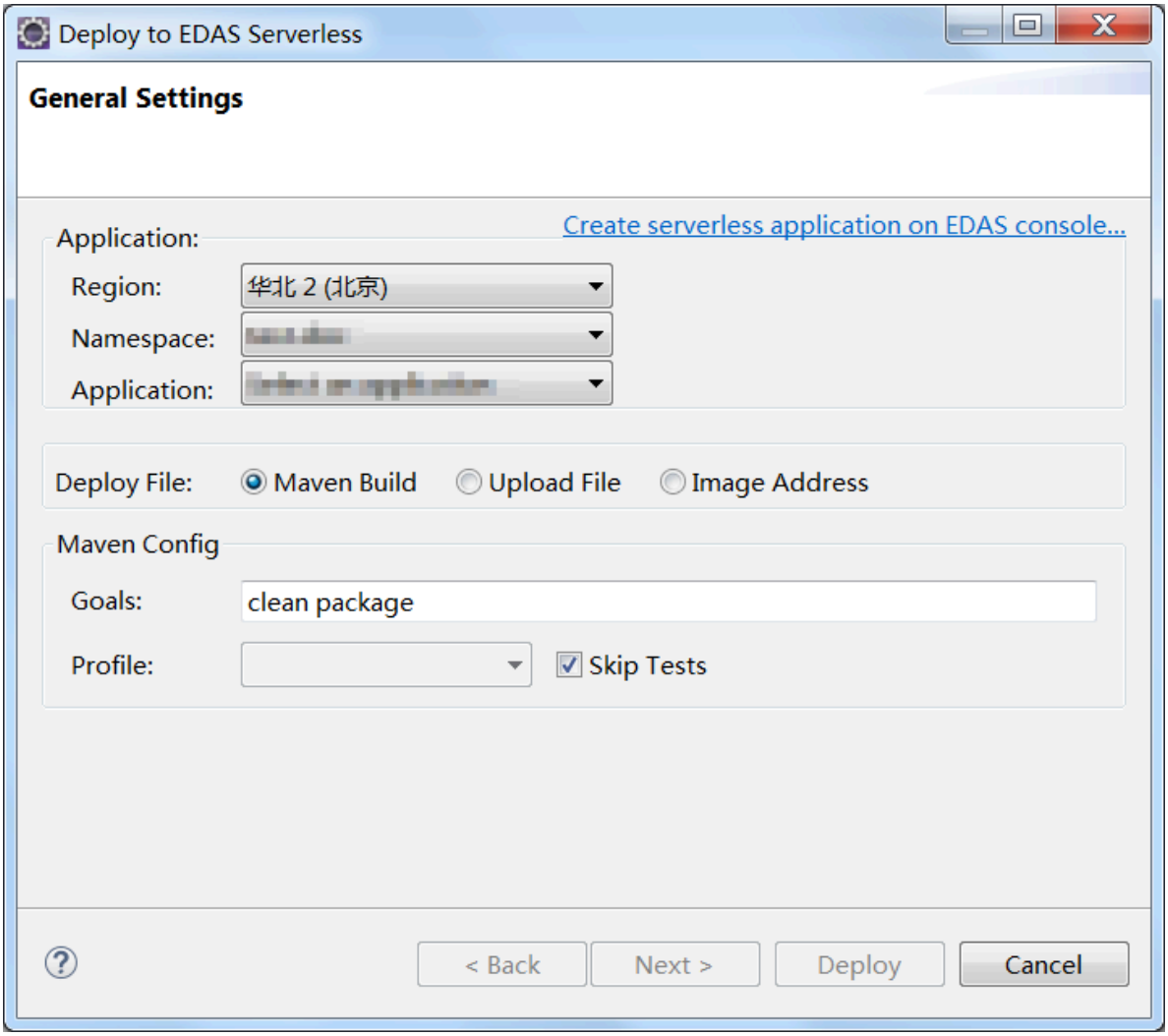
- 已有阿里云账号，在 Accounts 界面中单击 Get existing AK/SK，进入并登录阿里云登录页面，系统自动跳转至安全信息管理页面，获取 Access Key ID 和 Access Key Secret。
- 没有阿里云账号，在 Accounts 界面中单击 Sign up，进入阿里云账号注册页面并完成注册。注册完成后并获取 Access Key ID 和 Access Key Secret。

将应用部署到 SAE

Cloud Toolkit 插件目前仅支持将应用以 WAR包、JAR包或Image部署到 SAE。

1. 在 Eclipse 界面左侧的 Package Explorer 中选择创建的应用工程名，并在弹出的下拉菜单中选择Alibaba Cloud > Deploy to EDAS Serverless…。

2. 在 Deploy to EDAS Serverless 对话框中，依据需求选择应用的 Region、Namespace、Application，并设置部署方式，配置完成后单击 Deploy。



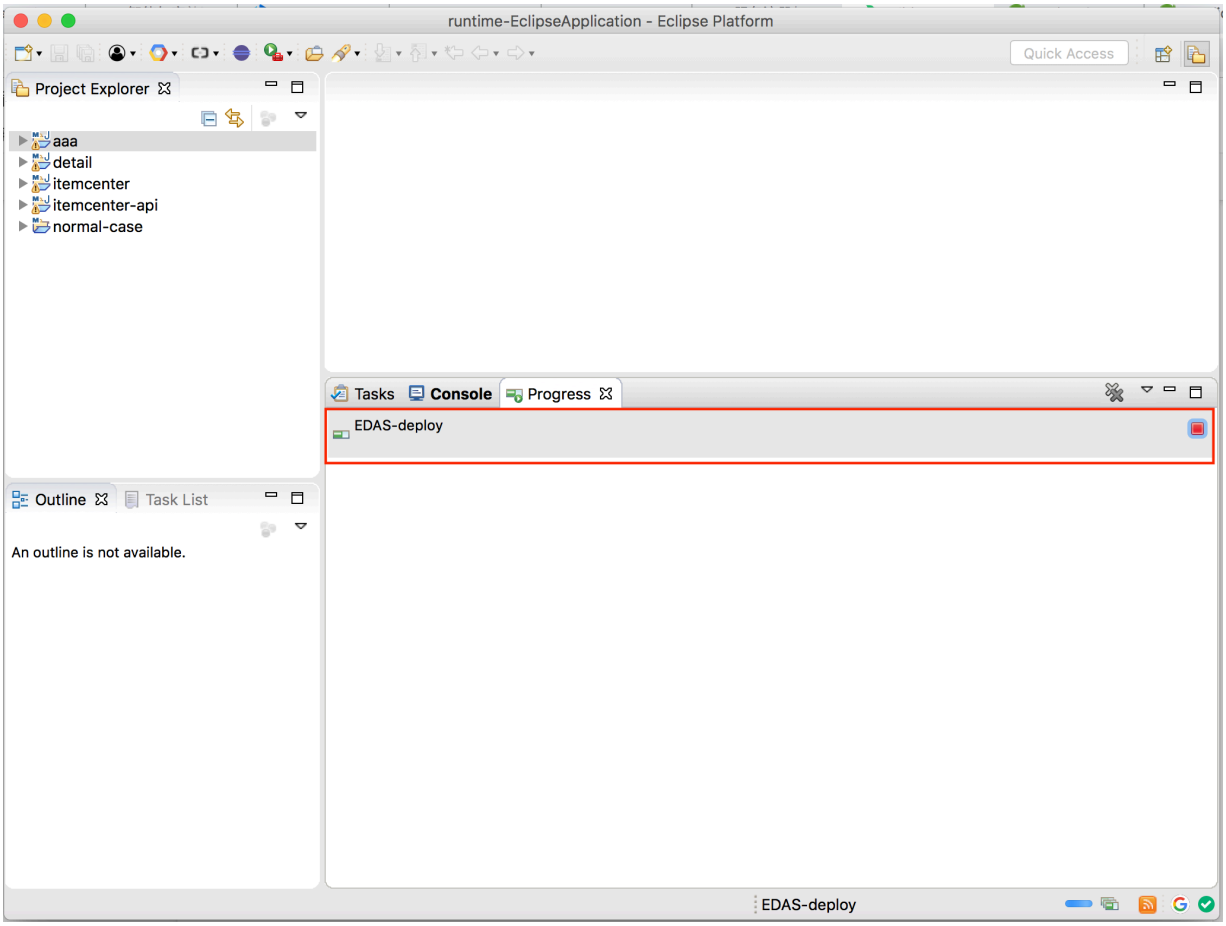
应用信息说明：

- Region：应用所在地域。
- Namespace：应用所在命名空间。
- Application：应用名称。

3. 部署过程中，Eclipse 的 Console 区域回显应用的部署日志，依据日志信息检查部署结果。

终止 Cloud Toolkit 插件运行

在插件运行过程中，现场需要插件运行，请在 Progress 页面终止 `EDAS-deploy` 进程。



问题反馈

如果您在使用 SAE 过程中有任何疑问，欢迎您扫描下面的二维码加入钉钉群进行反馈。

扫一扫群二维码，立刻加入该群。

4 高级设置

4.1 如何设置启动命令

使用 SAE 控制台进行镜像或者 Jar 包应用部署时，SAE 通过容器镜像或者Jar包中预设的参数启动容器。如果在启动前需要进行特殊配置（如Nginx），或者不想采用预设的启动参数，您可以在 SAE 设置容器启动命令进行特殊配置或者覆盖镜像的启动默认值。本配置适用于精通 Docker 的工程师。

背景信息

在制作镜像时容器的启动配置已经在 Dockerfile 文件中ENTRYPOINT 或 CMD 进行了配置，启动时所配置的内容会优先被执行。

例如 Dockerfile 中设置的 `ENTRYPOINT: [nginx, '-g', 'daemon off;']` 命令，在容器启动时被第一个执行。

```
FROM ubuntu
ENTRYPOINT [nginx, '-g', 'daemon off;']
```

设置镜像部署应用的启动命令

1. 登录 SAE 控制台。
2. 在左侧导航树选择Serverless 应用引擎 > 应用列表，在 应用列表页面右上角单击 创建应用。
3. 在应用的 应用基本信息页签内，设置应用相关信息并单击 下一步：应用部署配置。

4. 在 应用部署配置页面，选择应用部署方式为 镜像，并展开 启动命令设置并输入相关配置项。

例如设置命令如下：

- 启动命令：输入 `nginx`。
- 启动参数：输入`-g`。
- 单击 添加下一条，在新的参数行中增加输入参数 `daemon off` 。

> 注意：
>
> - 如果对原有 Dockerfile 镜像的 ENTRYPOINT 和 CMD 配置内容不熟悉，请勿自定义或者修改启动命令和启动参数，错误的启动命令将导致应用创建失败。
> - Docker 运行时仅支持一条 ENTRYPOINT 命令，SAE 控制台中设置的启动命令将会覆盖#unique_22时 Dockerfile 中所设置的 ENTRYPOINT 和 CMD 命令。

设置 Jar 包部署应用的启动命令

Java 应用启动和运行时需要定义启动自命令和参数，如 Java 的 JVM 参数、GC 策略等。

在 SAE 中，系统将上传的 Jar 包自动编译为镜像，并上传至镜像仓库，且以容器方式运行应用。在编译过程中，系统内置了启动命令和参数，指定了 Jar 包的存放路径。SAE 支持白频化修改 Jar 包默认启动命令及命令参数。

> 注意：
> 对原有 Dockerfile 镜像的 ENTRYPOINT 和 CMD 配置内容不熟悉，请勿自定义或者修改启动命令和启动参数，错误的启动命令将导致应用创建失败。

1. 登录 SAE 控制台。
2. 在左侧导航树选择Serverless 应用引擎 > 应用列表，在应用列表页面右上角单击创建应用。
3. 在应用的应用基本信息页签内，设置应用相关信息并单击下一步：应用部署配置。

4. 在应用部署配置页面，选择应用部署方式为Jar包部署，并展开启动命令设置输入相关配置项。

通过 options 设置线程的堆栈大小，通过 args 参数设置将标准输出和标准错误输出重新定向到指定文件。

4.2 如何设置环境变量

应用在系统中运行需要配置特定的环境变量，如 Java 应用程序在配置 Java_home 及其 Path 后其相关命令才得以执行。本文介绍如何为应用配置所需的环境变量。

操作指导

1. 登录 SAE 控制台。
2. 在左侧导航树选择Serverless 应用引擎 > 应用列表，并在 应用列表页面右上角单击 创建应用。
3. 在应用的 应用基本信息页签内，设置应用相关信息并单击 下一步：应用部署配置。
4. 在应用部署配置页面，展开环境变量设置并输入相关配置项。

   环境变量由环境变量名和环境变量值组成。 其内容可以包含大小写字母、数字和下划线（_），仅须以字母或下划线开始。字符长度建议不超过 256 个字符。

   说明:

   SAE 将环境变量作为属性存储在应用部署属性中，允许配置多个环境变量。

示例演示

某应用集成了 MySQL，在应用使用时需为其配置 MySQL 运行所需的环境变量。在通过 SAE 控制台进行部署时为其配置如下所示环境信息。

- MYSQL_ROOT_PASSWORD（必选项）： 用于设置 MySQL 的 Root 密码。不设置 MySQL 容器无法正常启动。

- MYSQL_USER 和 MYSQL_PASSWORD（可选项）：用于添加除 Root 之外的账号并设置密码，可选项。
- MYSQL_DATABASE（可选项）：用于设置生成容器时需要新建的数据库。

4.3 如何设置 Hosts 绑定

SAE 支持应用实例级别的实例，通过 Host 绑定对主机名进行解析，方便应用实例通过主机名进行访问。

操作步骤

1. 登录 SAE 控制台。
2. 在左侧导航树选择Serverless 应用引擎 > 应用列表，在 应用列表页面右上角单击 创建应用。
3. 在应用的 应用基本信息页签内，设置应用相关信息并单击 下一步：应用部署配置。
4. 在应用部署配置页面，展开 Hosts 绑定设置页签并输入 Host 配置项。

   如果当前应用已经部署在 SAE，请参见#unique_25/unique_25_Connect_42_section_3qn_6i4_g2p修改 Host 绑定。

5. 应用部署完成后，所设置的 Hosts 配置生效，可以通过域名访问应用。

4.4 如何设置应用健康检查

应用在 SAE 部署后，可以对应用进行健康检查，查看容器与业务能运行是否正常，方便异常时问题定位。

背景信息

健康检查是指由 Liveness 探针或者 Readiness 探针对容器与应用进行定时检查与汇报，并将结果反馈 SAE 控制台的过程。帮助您了解集群环境下整个服务的运行状态，方便问题定位。

SAE 基于Kubernetes，提供了如下两种健康检查方式。

- 应用实例存活检查（Liveness 配置）：针对单个应用实例进行健康检查，检测应用实例是否已经启动，已启动表示容器存活，反之亦然。如果容器存活检查失败，集群知会 SAE 对该容器进行重启；如果容器存活检查成功，表示应用运作正常，不执行任何操作。
- 应用业务就绪检查（Readiness 配置）：针对应用业务进行健康检查，检测用于处理客户请求的容器是否已经就绪。如果检测到容器未准备就绪，系统上报容器异常，不进行该容器的相关业务处理；如果容器准备就绪，则进行相关业务处理。

  此方式适用于启动时需要加载磁盘数据，或者依赖外部模块而导致启动时间长的业务。

操作指南

1. 登录 SAE 控制台。
2. 在左侧导航树选择Serverless 应用引擎 > 应用列表，在 应用列表页面右上角单击 创建应用。
3. 在应用的 应用基本信息页签内，设置应用相关信息并单击 下一步：应用部署配置。
4. 在应用部署配置页面，您可以展开应用健康检查设置页签并设置相关配置。

说明:

- 应用实例存活检查和应用业务就绪检查的参数相同。

- 在健康检查设置中应用实例存活检查（Liveness 配置）与应用业务就绪检查（Readiness 配置）二者可选配其一，也可二者均配。如果二者均配，SAE 首先进行应用实例存活检查，检查完成后进行应用业务就绪检查。

健康检查参数说明

- 延迟时间：输入延迟检测时间。例如设置为10，表示从应用启动后 10 秒开始检测。
- 超时时间：设置健康检查超时等待时间。例如设置为10，如果超时等待时间超过10秒，则本次健康检查失败，系统上报超时异常。若设置为0或不设置，默认超时等待时间为1秒。
- 执行命令：设置容器或者进程内部执行的健康检查命令。如果该命令退出状态码为0，表示容器健康 。

  命令格式请参见右侧的示例区域。

  举例：待执行命令`/run/server/-port=8080`，在 SAE 健康检擦进行如下设置。

5. 应用部署完成后，所设置的健康检查配置生效。

4.5 如何设置日志收集

SAE 实时日志功能支持查看500行日志信息，如果您有更高的查阅需求可以使用文件日志收集功能，系统将业务文件日志（不包含#stdout 和 stderr 日志）收集并输入 SLS 中，实现无限制行数查看日志、自行聚合分析日志，方便业务日志对接，按日志使用量计费。

前提条件

- 开通日志服务。
- 确保应用中每个实例预留了 0.25 核 CPU 和 25 MB Memory 的可用资源。

操作步骤

1. 登录 SAE 控制台。
2. 在左侧导航树选择轻量级分布式应用服务 > 应用列表，进入应用列表页面，单击右上角创建应用。
3. 在应用基本信息页签内，设置应用相关信息，配置完成后单击下一步：应用部署配置，并在应用部署配置页面，展开日志收集服务设置折叠页签。
4. 在日志收集服务设置折叠页签内，设置启用日志收集为 ON 。
5. 在收集规则配置区域中，输入日志源存放容器内的文件目录。

   配置多条收集规则，请单击添加依据实际情况进行设置。
6. 单击确认创建。

> 注意:
>
> SLS 日志服务账户内提供200个 Logstore 资源，50个 Project 资源。
>
> 应用开始创建后，系统自动检查日志服务是否开启、SLS 日志服务账号的内置资源是否足够。如果日志服务未开启，请依据提示开通日志服务；如果内置资源不足，请申请工单进行增额。

更多信息

应用创建或者重新部署完成后，SAE 依据所配的收集规则进行日志收集，您可以查看文件日志并进行相关业务分析。

5 制作应用容器 Docker 镜像

Spring Cloud 或 Dubbo 框架下编译的应用War 包或 Jar 包，如果需要在 SAE 上以镜像方式部署，请参见本文将应用的 War 包或 Jar 包制作为镜像并在 SAE 中发布。

注意事项

在制作应用镜像前，请查阅镜像创建规约并按照规约制作镜像。

创建标准 Dockerfile

Dockerfile 是以文本格式来快速创建自定义镜像的配置文件。

标准的Dockerfile描述了 SAE 创建应用运行环境所需的所有指令。

- 针对 War 包应用，该指令定义了对 JDK、Tomcat、War 包的下载、安装和启动等操作。
- 针对 Jar 包应用，该指令定义了对 JDK、Jar 包的下载、安装和启动等操作。

如下示例介绍如何制作不同框架应用镜像的 Dockerfile。

- HSF 应用的 Dockerfile 示例（基于 WAR 包）
- HSF 应用的 Dockerfile 示例（基于 JAR 包）
- Spring Cloud 或 Dubbo 应用的 Dockerfile 示例（基于 WAR 包）
- Spring Cloud 或 Dubbo 应用的 Dockerfile 示例（基于 JAR 包）

HSF 应用的 Dockerfile 示例（基于 WAR 包）

```
FROM centos:7
MAINTAINER 轻量级分布式应用服务 SAE 研发团队
# 安装打包必备软件
RUN yum -y install wget unzip telnet
# 准备 JDK/Tomcat 系统变量
ENV JAVA_HOME /usr/java/latest
ENV CATALINA_HOME /home/admin/apache-tomcat-7.0.91
ENV ADMIN_HOME /home/admin
ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/bin
RUN mkdir -p /home/admin
# 下载安装 JDK 7
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/agent/prod/files/
jdk-7u80-linux-x64.rpm -O /tmp/jdk-7u80-linux-x64.rpm && \
    yum -y install /tmp/jdk-7u80-linux-x64.rpm && \
    rm -rf /tmp/jdk-7u80-linux-x64.rpm
# 下载安装Tomcat
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/apache-tomcat-7.0
.91.tar.gz -O /tmp/apache-tomcat-7.0.91.tar.gz && \
    tar -xvf /tmp/apache-tomcat-7.0.91.tar.gz -C /home/admin && \
    rm /tmp/apache-tomcat-7.0.91.tar.gz && \
    chmod +x ${CATALINA_HOME}/bin/*sh

RUN mkdir -p ${CATALINA_HOME}/deploy/
# 下载部署 SAE 演示 WAR 包
```

```
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/demo/1.0/hello-
edas.war -O /tmp/ROOT.war && \
    unzip /tmp/ROOT.war -d ${CATALINA_HOME}/deploy/ROOT/ && \
    rm -rf /tmp/ROOT.war
# 设定 Tomcat 安装目录为容器启动目录，并采用 run 方式启动 Tomcat，在标准命令行输
出 catalina 日志
WORKDIR $ADMIN_HOME
CMD ["catalina.sh", "run"]
```

HSF 应用的 Dockerfile 示例（基于 JAR 包）

```
FROM centos:7
MAINTAINER 轻量级分布式应用服务 SAE 研发团队
# 安装打包必备软件
RUN yum -y install wget unzip telnet
# 准备 JDK/Tomcat 系统变量
ENV JAVA_HOME /usr/java/latest
ENV PATH $PATH:$JAVA_HOME/bin
ENV ADMIN_HOME /home/admin
# 下载安装 JDK 7
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/agent/prod/files/
jdk-7u80-linux-x64.rpm -O /tmp/jdk-7u80-linux-x64.rpm && \
    yum -y install /tmp/jdk-7u80-linux-x64.rpm && \
    rm -rf /tmp/jdk-7u80-linux-x64.rpm
# 下载部署 SAE 演示 JAR 包
RUN mkdir -p /home/admin/app/ && \
         wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/demo/1.0
/hello-edas-0.0.1-SNAPSHOT.jar -O /home/admin/app/hello-edas-0.0.1-
SNAPSHOT.jar
# 将启动命令写入启动脚本 start.sh
RUN mkdir -p /home/admin
RUN echo '$JAVA_HOME/bin/java -jar $CATALINA_OPTS /home/admin/app/
hello-edas-0.0.1-SNAPSHOT.jar'> /home/admin/start.sh && chmod +x /home
/admin/start.sh
WORKDIR $ADMIN_HOME
CMD ["/bin/bash", "/home/admin/start.sh"]
```

Spring Cloud 或 Dubbo 应用的 Dockerfile 示例（基于 WAR 包）

```
FROM centos:7
MAINTAINER 轻量级分布式应用服务 SAE 研发团队

# 安装打包必备软件
RUN yum -y install wget unzip telnet

# 准备 JDK/Tomcat 系统变量
ENV JAVA_HOME /usr/java/latest
ENV CATALINA_HOME /home/admin/apache-tomcat-7.0.91
ENV ADMIN_HOME /home/admin
ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/bin
RUN mkdir -p /home/admin

# 下载安装 OpenJDK
RUN yum -y install java-1.8.0-openjdk-devel

# 下载安装 Tomcat
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/apache-tomcat-7.0
.91.tar.gz -O /tmp/apache-tomcat-7.0.91.tar.gz && \
    tar -xvf /tmp/apache-tomcat-7.0.91.tar.gz -C /home/admin && \
    rm /tmp/apache-tomcat-7.0.91.tar.gz && \
```

```
    chmod +x ${CATALINA_HOME}/bin/*sh

RUN mkdir -p ${CATALINA_HOME}/deploy/

# 增加容器内中文支持
ENV LANG="en_US.UTF-8"

# 增强 Webshell 使用体验
ENV TERM=xterm

# 下载部署 SAE 演示 WAR 包
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/demo/1.0/hello-
edas.war -O /tmp/ROOT.war && \
    unzip /tmp/ROOT.war -d ${CATALINA_HOME}/deploy/ROOT/ && \
    rm -rf /tmp/ROOT.war

# 设定 Tomcat 安装目录为容器启动目录，并采用 run 方式启动 Tomcat，在标准命令行输
出 catalina 日志
WORKDIR $ADMIN_HOME
CMD ["catalina.sh", "run"]
```

Spring Cloud 或 Dubbo 应用的 Dockerfile 示例（基于 JAR 包）

```
FROM centos:7
MAINTAINER 轻量级分布式应用服务 SAE 研发团队

# 安装打包必备软件
RUN yum -y install wget unzip telnet

# 准备 JDK/Tomcat 系统变量
ENV JAVA_HOME /usr/java/latest
ENV PATH $PATH:$JAVA_HOME/bin
ENV ADMIN_HOME /home/admin

# 下载安装 OpenJDK
RUN yum -y install java-1.8.0-openjdk-devel

# 下载部署 SAE 演示 JAR 包
RUN mkdir -p /home/admin/app/ && \
         wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/demo/1.0
/hello-edas-0.0.1-SNAPSHOT.jar -O /home/admin/app/hello-edas-0.0.1-
SNAPSHOT.jar

# 增加容器内中文支持
ENV LANG="en_US.UTF-8"

# 增强 Webshell 使用体验
ENV TERM=xterm

# 将启动命令写入启动脚本 start.sh
RUN mkdir -p /home/admin
RUN echo '$JAVA_HOME/bin/java -jar $CATALINA_OPTS /home/admin/app/
hello-edas-0.0.1-SNAPSHOT.jar'> /home/admin/start.sh && chmod +x /home
/admin/start.sh
WORKDIR $ADMIN_HOME
CMD ["/bin/bash", "/home/admin/start.sh"]
```

自定义设置 Dockerfile

通过编辑 Dockerfile 方式进行运行环境配置修改，如 JDK 版本替换、Tomcat 配置修改、更改运行时环境等操作。

- 更换 JDK 版本

  在标准 Dockerfile 文件中，请参见如下示例更换其他版本的 JDK。

```
# 下载安装 JDK 8
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/agent/prod/
files/jdk-8u65-linux-x64.rpm -O /tmp/jdk-8u65-linux-x64.rpm && \
    yum -y install /tmp/jdk-8u65-linux-x64.rpm && \
    rm -rf /tmp/jdk-8u65-linux-x64.rpm
```

- 在 Tomcat 启动参数中添加 SAE 运行时环境

  SAE 提供了 JVM 环境变量 `DEAS_CATALINA_OPTS` 包含运行所需的基本参数。此外 Ali-Tomcat 提供了自定义 JVM 参数配置选项 `JAVA_OPTS`，可以设置 Xmx Xms 等参数。 启动参数具体详情请参见应用运行时环境变量 ,

```
# 设置 SAE 应用 JVM 参数
ENV CATALINA_OPTS ${EDAS_CATALINA_OPTS}
# 设置 JVM 参数
ENV JAVA_OPTS="\
     -Xmx3550m \
     -Xms3550m \
     -Xmn2g \
     -Xss128k"
```

在本地构建镜像

从本地命令行进入 Dockerfile 所在的目录，执行 docker build 命令构建镜像：

```
docker build -t [标签名称，最好取应用名]:[版本号] .
或
docker build -t [标签名称，最好取应用名]:[版本号] -f /path/to/custom_doc
kerfile_name .  #假如您创建好的 Dockerfile 在其他位置或名称不为 Dockerfile
 时适用。
```

列如：

```
docker build -t hsf-provider:1.0.0 .
```

构建完成后，使用 `docker images | grep <镜像标签名称>` 命令查看本地编译好的镜像。

上传镜像到镜像仓库

上传本地编译好的镜像到镜像仓库，并在 SAE 中选择该镜像进行部署。

例如：推送镜像 registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20181111 镜像到镜像仓库。

```
docker push registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20181111
```

镜像创建规约

通过 Dockerfile 制作自定义镜像时，须遵循以下规范及限制。

- 租户/加密信息

  SAE 应用用户鉴权与加密凭证的信息。

  表 5-1: 环境变量

环境变量 Key	类型	描述
tenantId	String	SAE 租户 ID
accessKey	String	鉴权 Access Key ID
secretKey	String	鉴权 Access Key Secret

  表 5-2: 本地文件

路径	类型	描述
/home/admin/.spas_key/default	File	SAE 租户鉴权信息，包含上述 ENV 信息

- 服务信息

  包含运行时所需连接的 SAE 域名、端口等信息。

  表 5-3: 环境变量

环境变量 Key	描述
EDAS_ADDRESS_SERVER_DOMAIN	配置中心服务域名或 IP
EDAS_ADDRESS_SERVER_PORT	配置中心服务端口
EDAS_CONFIGSERVER_CLIENT_PORT	CS 服务端口

- 应用运行时环境变量（强制）

  SAE 部署时会提供以下环境变量，为保证应用运行正常，请勿覆盖配置。

环境变量 Key	描述
POD_IP	POD IP
EDAS_APP_ID	SAE 应用 ID
EDAS_ECC_ID	SAE ECC ID
EDAS_PROJECT_NAME	同 EDAS_APP_ID，用于调用链解析
EDAS_JM_CONTAINER_ID	同 EDAS_ECC_ID，用于调用链解析
EDAS_CATALINA_OPTS	中间件运行时所需 CATALINA_OPTS 参数
CATALINA_OPTS	同 EDAS_CATALINA_OPTS，默认 TOMCAT 启动参数