Red Hat build of MicroShift 4.16

CLI 工具

了解如何使用 MicroShift 的命令行工具

Last Updated: 2024-07-02

Red Hat build of MicroShift 4.16 CLI 工具

了解如何使用 MicroShift 的命令行工具

法律通告

摘要

本文档提供有关为 MicroShift 使用命令行工具的信息。详细列出了安装和配置可选 CLI 工具，如 'oc' 和 'kubectl'。还包括 CLI 命令参考和使用方法的示例。

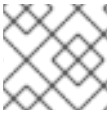
目录

第 1 章 红帽构建的 MICROSHIFT CLI 工具简介

您可以使用不同的命令行界面(CLI)工具来构建、部署和管理 MicroShift 集群和工作负载的红帽构建。使用 CLI 工具，您可以从终端完成各种管理和开发操作，以管理部署并与系统的每个组件交互。

用于红帽构建的 MicroShift 的 CLI 工具如下：

- Kubernetes CLI (**kubectl**)
- 带有启用的命令子集的 OpenShift CLI (**oc**) 工具
- 内置 **microshift** 命令类型



注意

红帽构建的 MicroShift 不支持多节点部署、项目和开发工具的命令。

1.1. 其他资源

- 为 MicroShift 安装 OpenShift CLI 工具。
- OpenShift CLI (oc)的详细描述。
- 用于特定用例的 Red Hat Enterprise Linux (RHEL) 文档 。
- 使用 kubeconfig 进行集群访问

第 2 章 OPENSHIFT CLI 入门

要使用 OpenShift CLI (**oc**) 工具，您必须独立于您的 MicroShift 安装下载并安装它。

2.1. 安装 OPENSHIFT CLI

您可以通过下载二进制文件或使用 Homebrew 来安装 OpenShift CLI (**oc**)。

2.1.1. 安装 OpenShift CLI

您可以安装 OpenShift CLI(**oc**)来使用命令行界面与红帽构建的 MicroShift 进行交互。您可以在 Linux、Windows 或 macOS 上安装 **oc**。

重要

如果安装了旧版本的 **oc**，则无法使用红帽构建 MicroShift 4.16 中的所有命令。下载并安装新版本的 **oc**。

在 Linux 上安装 OpenShift CLI
您可以按照以下流程在 Linux 上安装 OpenShift CLI(**oc**)二进制文件。

注意

红帽构建的 MicroShift 版本编号与 OpenShift Container Platform 版本号匹配。使用与 MicroShift 版本匹配的 **oc** 二进制文件，并具有适当的 RHEL 兼容性。

流程

1. 导航到红帽客户门户网站上的 OpenShift Container Platform 下载页面。
2. 从 产品变体 下拉列表中选择架构。
3. 从 版本 下拉列表中选择适当的版本。
4. 点 **OpenShift v4.16 Linux Client** 条目旁的 **Download Now** 来保存文件。
5. 解包存档：

```
$ tar xvf <file>
```

6. 将 **oc** 二进制文件放到 **PATH** 中的目录中。
要查看您的 **PATH**，请执行以下命令：

```
$ echo $PATH
```

验证

- 安装 OpenShift CLI 后，可以使用 **oc** 命令：

```
$ oc <command>
```

在 Windows 上安装 OpenShift CLI

您可以按照以下流程在 Windows 上安装 OpenShift CLI(**oc**)二进制文件。

注意

红帽构建的 MicroShift 版本编号与 OpenShift Container Platform 版本号匹配。使用与 MicroShift 版本匹配的 **oc** 二进制文件，并具有适当的 RHEL 兼容性。

流程

1. 导航到红帽客户门户网站上的 OpenShift Container Platform 下载页面。
2. 从 版本 下拉列表中选择适当的版本。
3. 点 **OpenShift v4.16 Windows Client** 条目旁的 **Download Now** 来保存文件。
4. 使用 ZIP 程序解压存档。
5. 将 **oc** 二进制文件移到 **PATH** 中的目录中。
   要查看您的 **PATH**，请打开命令提示并执行以下命令：

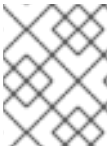
```
C:\> path
```

验证

- 安装 OpenShift CLI 后，可以使用 **oc** 命令：

```
C:\> oc <command>
```

在 macOS 上安装 OpenShift CLI
您可以按照以下流程在 macOS 上安装 OpenShift CLI(**oc**)二进制文件。



注意

红帽构建的 MicroShift 版本编号与 OpenShift Container Platform 版本号匹配。使用与 MicroShift 版本匹配的 **oc** 二进制文件，并具有适当的 RHEL 兼容性。

流程

1. 导航到红帽客户门户网站上的 OpenShift Container Platform 下载页面。
2. 从 版本 下拉列表中选择适当的版本。
3. 点 **OpenShift v4.16 macOS Client** 条目旁的 **Download Now** 来保存文件。
4. 解包和解压存档。
5. 将 **oc** 二进制文件移到 PATH 的目录中。
   要查看您的 **PATH**，请打开终端并执行以下命令：

```
$ echo $PATH
```

验证

- 安装 OpenShift CLI 后，可以使用 **oc** 命令：

```
$ oc <command>
```

2.1.2. 使用 Homebrew 安装 OpenShift CLI

对于 macOS，您可以使用 Homebrew 软件包管理器安装 OpenShift CLI(**oc**)。

先决条件

- 已安装 Homebrew(**brew**)。

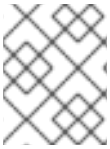
流程

- 运行以下命令来安装 openshift-cli 软件包：

```
$ brew install openshift-cli
```

2.1.3. 使用 RPM 安装 OpenShift CLI

对于 Red Hat Enterprise Linux (RHEL)，如果您的红帽帐户中包括有效的红帽构建的 MicroShift 订阅，则可将通过 RPM 安装 OpenShift CLI (**oc**)。



注意

不支持将 OpenShift CLI(**oc**)安装为 Red Hat Enterprise Linux(RHEL)9 的 RPM。您必须下载二进制文件，为 RHEL 9 安装 OpenShift CLI。

先决条件

- 必须具有 root 或 sudo 权限。

流程

1. 使用 Red Hat Subscription Manager 注册：

```
# subscription-manager register
```

2. 获取最新的订阅数据：

```
# subscription-manager refresh
```

3. 列出可用的订阅：

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 在上一命令的输出中，找到红帽构建的 MicroShift 订阅的池 ID，并把订阅附加到注册的系统：

```
# subscription-manager attach --pool=<pool_id>
```

5. 启用红帽构建 MicroShift 4.16 所需的存储库。

```
# subscription-manager repos --enable="rhocp-4.16-for-rhel-8-x86_64-rpms"
```

6. 安装 **openshift-clients** 软件包：

```
# yum install openshift-clients
```

安装 CLI 后，就可以使用**oc**命令：

```
$ oc <command>
```

第 3 章 配置 OPENSHIFT CLI

根据您的偏好配置 **oc** 以使用它。

3.1. 启用 TAB 自动完成功能

您可以为 Bash 或 Zsh shell 启用 tab 自动完成功能。

3.1.1. 为 Bash 启用 tab 自动完成

安装 OpenShift CLI (**oc**)后，您可以启用 tab 自动完成功能，以便在按 Tab 键时自动完成 **oc** 命令或建议选项。以下流程为 Bash shell 启用 tab 自动完成功能。

先决条件

- 已安装 OpenShift CLI (**oc**)。
- 已安装软件包 **bash-completion**。

流程

1. 将 Bash 完成代码保存到一个文件中：

```
$ oc completion bash > oc_bash_completion
```

2. 将文件复制到 **/etc/bash_completion.d/**：

```
$ sudo cp oc_bash_completion /etc/bash_completion.d/
```

   您也可以将文件保存到一个本地目录，并从您的**.bashrc**文件中 source 这个文件。

开新终端时 tab 自动完成功能将被启用。

3.1.2. 为 Zsh 启用 tab 自动完成功能

安装 OpenShift CLI (**oc**)后，您可以启用 tab 自动完成功能，以便在按 Tab 键时自动完成 **oc** 命令或建议选项。以下流程为 Zsh shell 启用 tab 自动完成功能。

先决条件

- 已安装 OpenShift CLI (**oc**)。

流程

- 要在 **.zshrc** 文件中为 **oc** 添加 tab 自动完成功能，请运行以下命令：

```
$ cat >>~/.zshrc<<EOF
autoload -Uz compinit
compinit
if [ $commands[oc] ]; then
  source <(oc completion zsh)
```

```
  compdef _oc oc
fi
EOF
```

开新终端时 tab 自动完成功能将被启用。

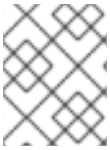
第 4 章 使用 OC 工具

可选的 OpenShift CLI (**oc**)工具为红帽构建的 MicroShift 部署提供了 **oc** 命令的子集。如果您熟悉 OpenShift Container Platform 和 Kubernetes，则使用 **oc** 非常方便。

4.1. 关于 OPENSHIFT CLI

借助 OpenShift 命令行界面 (CLI)，**oc** 命令可从终端部署和管理 MicroShift 项目。CLI **oc** 工具在以下情况下是理想的选择：

- 直接使用项目源代码
- 编写红帽构建的 MicroShift 操作
- 管理项目，同时受带宽资源的限制



注意

kubeconfig 文件必须存在，集群才能访问。这些值从内置默认值或 **config.yaml** 应用（如果创建了）。

4.2. 在 RED HAT BUILD OF MICROSHIFT 中使用 OPENSHIFT CLI

查看以下部分以了解如何使用 **oc** CLI 在红帽构建的 MicroShift 中完成常见任务。

4.2.1. 查看 pod

使用**oc get pods**命令查看当前项目的 pod。

注意

当您在 pod 中运行 **oc** 且没有指定命名空间时，默认使用 pod 的命名空间。

```
$ oc get pods -o wide
```

输出示例

```
NAME                READY   STATUS      RESTARTS   AGE     IP           NODE
NOMINATED NODE
cakephp-ex-1-build  0/1     Completed   0          5m45s   10.131.0.10  ip-10-0-141-74.ec2.internal
<none>
cakephp-ex-1-deploy 0/1     Completed   0          3m44s   10.129.2.9   ip-10-0-147-65.ec2.internal
<none>
cakephp-ex-1-ktz97  1/1     Running     0          3m33s   10.128.2.11  ip-10-0-168-105.ec2.internal
<none>
```

4.2.2. 查看 pod 日志

使用**oc logs**命令查看特定 pod 的日志。

```
$ oc logs cakephp-ex-1-deploy
```

输出示例

```
--> Scaling cakephp-ex-1 to 1
--> Success
```

4.2.3. 列出支持的 API 资源

使用**oc api-resources**命令查看服务器上支持的 API 资源列表。

```
$ oc api-resources
```

输出示例

```
NAME                              SHORTNAMES   APIGROUP                       NAMESPACED   KIND
bindings                                                                      true         Binding
componentstatuses                 cs                                          false        ComponentStatus
configmaps                        cm                                          true         ConfigMap
...
```

4.3. 获得帮助

您可以通过以下方式获得 CLI 命令和 MicroShift 资源的帮助：

- 使用 **oc help --flag** 来获取有关特定 CLI 命令的信息：

  示例：获取oc create命令的帮助信息

  ```
  $ oc create --help
  ```

  输出示例

  ```
  Create a resource by filename or stdin

  JSON and YAML formats are accepted.

  Usage:
    oc create -f FILENAME [flags]

  ...
  ```

- 使用**oc explain**命令查看特定资源的描述信息和项信息：

  示例：查看 Pod 资源的文档

  ```
  $ oc explain pods
  ```

  输出示例

  ```
  KIND:     Pod
  VERSION:  v1
  ```

```
DESCRIPTION:
    Pod is a collection of containers that can run on a host. This resource is
    created by clients and scheduled onto hosts.

FIELDS:
  apiVersion <string>
    APIVersion defines the versioned schema of this representation of an
    object. Servers should convert recognized schemas to the latest internal
    value, and may reject unrecognized values. More info:
    https://git.k8s.io/community/contributors/devel/api-conventions.md#resources

...
```

4.4. 红帽构建的 MICROSHIFT 中的 OC 命令错误

并非所有 OpenShift CLI (oc) 工具命令都与红帽构建的 MicroShift 部署相关。当您使用 **oc** 对不受支持的 API 发出请求调用时，**oc** 二进制文件通常会生成有关无法找到资源的错误消息。

输出示例

例如，当运行以下 **new-project** 命令时：

```
$ oc new-project test
```

可能会生成以下出错信息：

```
Error from server (NotFound): the server could not find the requested resource (get
projectrequests.project.openshift.io)
```

当 **get projects** 命令运行时，可以会产生另一个错误，如下所示：

```
$ oc get projects
error: the server doesn't have a resource type "projects"
```

第 5 章 使用 OC 和 KUBECTL 命令

Kubernetes 命令行界面（CLI）**kubectl** 可以用来对 Kubernetes 集群运行命令。因为 Red Hat build of MicroShift 是一个经过认证的 Kubernetes 发行版本，所以您可以使用红帽构建的 MicroShift 附带的受支持的 **kubectl** CLI 工具，或使用 **oc** CLI 工具获得扩展功能。

5.1. KUBECTL CLI 工具

您可以使用 **kubectl** CLI 工具与红帽构建的 MicroShift 集群上的 Kubernetes 原语交互。您还可以将现有 **kubectl** 工作流和脚本用于来自另一个 Kubernetes 环境的新红帽构建的 MicroShift 用户，或适用于希望使用 **kubectl** CLI 的用户。

如果您下载 **oc** CLI 工具，则 **kubectl** CLI 工具会包含在存档中。

如需更多信息，请参阅 Kubernetes CLI 工具文档。

5.2. OC CLI 工具

oc CLI 工具提供与 **kubectl** CLI 工具相同的功能，但它被扩展为原生支持额外的红帽构建 MicroShift 功能，包括：

- 路由资源
  Route 资源对象特定于红帽构建的 MicroShift 发行版本，基于标准 Kubernetes 原语构建。
- 附加命令
  例如，借助附加命令 **oc new-app** 可以更轻松地使用现有源代码或预构建镜像来启动新的应用程序。

重要

如果安装了旧版本的 **oc** CLI 工具，则无法使用它完成红帽构建的 MicroShift 4.16 中的所有命令。如果需要最新的功能，您必须下载并安装与红帽构建的 MicroShift 版本对应的 **oc** CLI 工具的最新版本。

非安全 API 更改至少涉及两个次发行版本（例如，4.1 到 4.2 到 4.3）来更新旧的 **oc** 二进制文件。使用新功能可能需要较新的 **oc** 二进制文件。一个 4.3 服务器可能会带有版本 4.2 **oc** 二进制文件无法使用的功能，而一个 4.3 **oc** 二进制文件可能会带有 4.2 服务器不支持的功能。

表 5.1. 兼容性列表

	X.Y (**oc** Client)	**X.Y+N** footnote:versionpolicyn[其中 **N** 是一个大于或等于 1 的数字] (**oc** Client)
X.Y (Server)	1	3
X.Y+N footnote:versionpolicyn[] (Server)	2	1

1 完全兼容。

2 **oc** 客户端可能无法访问服务器的功能。

3 **oc** 客户端可能会提供与要访问的服务器不兼任的选项和功能。

第 6 章 OPENSHIFT CLI 命令参考

OpenShift CLI (**oc**) 命令的描述和示例命令包括在此参考文档中。您必须具有 **cluster-admin** 或同等权限才能使用这些命令。要列出管理员命令及其信息，请使用以下命令：

- 输入 **oc adm -h** 命令来列出所有管理员命令：

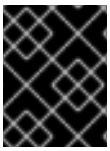
  命令语法

  ```
  $ oc adm -h
  ```

- 输入 **oc <command> --help** 命令以获取特定命令的更多详情：

  命令语法

  ```
  $ oc <command> --help
  ```



重要

使用 **oc <command> --help** 列出所有 **oc** 命令的详情。并非所有 **oc** 命令都适用于使用 MicroShift 的红帽构建。

6.1. OPENSHIFT CLI（OC）开发人员命令

6.1.1. oc annotate

更新资源上的注解

用法示例

```
  # Update pod 'foo' with the annotation 'description' and the value 'my frontend'
  # If the same annotation is set multiple times, only the last value will be applied
  oc annotate pods foo description='my frontend'

  # Update a pod identified by type and name in "pod.json"
  oc annotate -f pod.json description='my frontend'

  # Update pod 'foo' with the annotation 'description' and the value 'my frontend running nginx',
overwriting any existing value
  oc annotate --overwrite pods foo description='my frontend running nginx'

  # Update all pods in the namespace
  oc annotate pods --all description='my frontend running nginx'

  # Update pod 'foo' only if the resource is unchanged from version 1
  oc annotate pods foo description='my frontend running nginx' --resource-version=1

  # Update pod 'foo' by removing an annotation named 'description' if it exists
  # Does not require the --overwrite flag
  oc annotate pods foo description-
```

6.1.2. oc api-resources

在服务器上显示支持的 API 资源

用法示例

```
# Print the supported API resources
oc api-resources

# Print the supported API resources with more information
oc api-resources -o wide

# Print the supported API resources sorted by a column
oc api-resources --sort-by=name

# Print the supported namespaced resources
oc api-resources --namespaced=true

# Print the supported non-namespaced resources
oc api-resources --namespaced=false

# Print the supported API resources with a specific APIGroup
oc api-resources --api-group=rbac.authorization.k8s.io
```

6.1.3. oc api-versions

以"group/version"的形式输出服务器上支持的 API 版本。

用法示例

```
# Print the supported API versions
oc api-versions
```

6.1.4. oc apply

通过文件名或 stdin 将配置应用到资源

用法示例

```
# Apply the configuration in pod.json to a pod
oc apply -f ./pod.json

# Apply resources from a directory containing kustomization.yaml - e.g. dir/kustomization.yaml
oc apply -k dir/

# Apply the JSON passed into stdin to a pod
cat pod.json | oc apply -f -

# Apply the configuration from all files that end with '.json' - i.e. expand wildcard characters in file names
oc apply -f '*.json'

# Note: --prune is still in Alpha
# Apply the configuration in manifest.yaml that matches label app=nginx and delete all other resources that are not in the file and match label app=nginx
oc apply --prune -f manifest.yaml -l app=nginx
```

```
# Apply the configuration in manifest.yaml and delete all the other config maps that are not in the file
oc apply --prune -f manifest.yaml --all --prune-allowlist=core/v1/ConfigMap
```

6.1.5. oc apply edit-last-applied

编辑资源/对象的最新 last-applied-configuration 注解

用法示例

```
# Edit the last-applied-configuration annotations by type/name in YAML
oc apply edit-last-applied deployment/nginx

# Edit the last-applied-configuration annotations by file in JSON
oc apply edit-last-applied -f deploy.yaml -o json
```

6.1.6. oc apply set-last-applied

设置 live 对象上的 last-applied-configuration 注释，以匹配文件的内容。

用法示例

```
# Set the last-applied-configuration of a resource to match the contents of a file
oc apply set-last-applied -f deploy.yaml

# Execute set-last-applied against each configuration file in a directory
oc apply set-last-applied -f path/

# Set the last-applied-configuration of a resource to match the contents of a file; will create the annotation if it does not already exist
oc apply set-last-applied -f deploy.yaml --create-annotation=true
```

6.1.7. oc apply view-last-applied

查看资源/对象最新的最后应用配置注解

用法示例

```
# View the last-applied-configuration annotations by type/name in YAML
oc apply view-last-applied deployment/nginx

# View the last-applied-configuration annotations by file in JSON
oc apply view-last-applied -f deploy.yaml -o json
```

6.1.8. oc attach

附加到正在运行的容器

用法示例

```
# Get output from running pod mypod; use the 'oc.kubernetes.io/default-container' annotation
# for selecting the container to be attached or the first container in the pod will be chosen
```

```
oc attach mypod

# Get output from ruby-container from pod mypod
oc attach mypod -c ruby-container

# Switch to raw terminal mode; sends stdin to 'bash' in ruby-container from pod mypod
# and sends stdout/stderr from 'bash' back to the client
oc attach mypod -c ruby-container -i -t

# Get output from the first pod of a replica set named nginx
oc attach rs/nginx
```

6.1.9. oc auth can-i

检查是否允许操作

用法示例

```
# Check to see if I can create pods in any namespace
oc auth can-i create pods --all-namespaces

# Check to see if I can list deployments in my current namespace
oc auth can-i list deployments.apps

# Check to see if service account "foo" of namespace "dev" can list pods
# in the namespace "prod".
# You must be allowed to use impersonation for the global option "--as".
oc auth can-i list pods --as=system:serviceaccount:dev:foo -n prod

# Check to see if I can do everything in my current namespace ("*" means all)
oc auth can-i '*' '*'

# Check to see if I can get the job named "bar" in namespace "foo"
oc auth can-i list jobs.batch/bar -n foo

# Check to see if I can read pod logs
oc auth can-i get pods --subresource=log

# Check to see if I can access the URL /logs/
oc auth can-i get /logs/

# List all allowed actions in namespace "foo"
oc auth can-i --list --namespace=foo
```

6.1.10. oc auth reconcile

协调 RBAC 角色、角色绑定、集群角色和集群角色绑定对象的规则

用法示例

```
# Reconcile RBAC resources from a file
oc auth reconcile -f my-rbac-rules.yaml
```

6.1.11. oc auth whoami

实验性：检查自我主题属性

用法示例

```
# Get your subject attributes.
oc auth whoami

# Get your subject attributes in JSON format.
oc auth whoami -o json
```

6.1.12. oc cluster-info

显示集群信息

用法示例

```
# Print the address of the control plane and cluster services
oc cluster-info
```

6.1.13. oc cluster-info dump

转储用于调试和诊断的相关信息

用法示例

```
# Dump current cluster state to stdout
oc cluster-info dump

# Dump current cluster state to /path/to/cluster-state
oc cluster-info dump --output-directory=/path/to/cluster-state

# Dump all namespaces to stdout
oc cluster-info dump --all-namespaces

# Dump a set of namespaces to /path/to/cluster-state
oc cluster-info dump --namespaces default,kube-system --output-directory=/path/to/cluster-state
```

6.1.14. oc completion

输出指定 shell 的 shell 完成代码 (bash、zsh、fish 或 powershell)

用法示例

```
# Installing bash completion on macOS using homebrew
## If running Bash 3.2 included with macOS
brew install bash-completion
## or, if running Bash 4.1+
brew install bash-completion@2
## If oc is installed via homebrew, this should start working immediately
## If you've installed via other means, you may need add the completion to your completion directory
oc completion bash > $(brew --prefix)/etc/bash_completion.d/oc
```

```
# Installing bash completion on Linux
## If bash-completion is not installed on Linux, install the 'bash-completion' package
## via your distribution's package manager.
## Load the oc completion code for bash into the current shell
source <(oc completion bash)
## Write bash completion code to a file and source it from .bash_profile
oc completion bash > ~/.kube/completion.bash.inc
printf "
# Kubectl shell completion
source '$HOME/.kube/completion.bash.inc'
" >> $HOME/.bash_profile
source $HOME/.bash_profile

# Load the oc completion code for zsh[1] into the current shell
source <(oc completion zsh)
# Set the oc completion code for zsh[1] to autoload on startup
oc completion zsh > "${fpath[1]}/_oc"


# Load the oc completion code for fish[2] into the current shell
oc completion fish | source
# To load completions for each session, execute once:
oc completion fish > ~/.config/fish/completions/oc.fish

# Load the oc completion code for powershell into the current shell
oc completion powershell | Out-String | Invoke-Expression
# Set oc completion code for powershell to run on startup
## Save completion code to a script and execute in the profile
oc completion powershell > $HOME\.kube\completion.ps1
Add-Content $PROFILE "$HOME\.kube\completion.ps1"
## Execute completion code in the profile
Add-Content $PROFILE "if (Get-Command oc -ErrorAction SilentlyContinue) {
oc completion powershell | Out-String | Invoke-Expression
}"
## Add completion code directly to the $PROFILE script
oc completion powershell >> $PROFILE
```

6.1.15. oc config current-context

显示 current-context

用法示例

```
# Display the current-context
oc config current-context
```

6.1.16. oc config delete-cluster

从 kubeconfig 删除指定的集群

用法示例

```
# Delete the minikube cluster
oc config delete-cluster minikube
```

6.1.17. oc config delete-context

从 kubeconfig 删除指定的上下文

用法示例

```
# Delete the context for the minikube cluster
oc config delete-context minikube
```

6.1.18. oc config delete-user

从 kubeconfig 删除指定用户

用法示例

```
# Delete the minikube user
oc config delete-user minikube
```

6.1.19. oc config get-clusters

显示 kubeconfig 中定义的集群

用法示例

```
# List the clusters that oc knows about
oc config get-clusters
```

6.1.20. oc config get-contexts

描述一个或多个上下文

用法示例

```
# List all the contexts in your kubeconfig file
oc config get-contexts

# Describe one context in your kubeconfig file
oc config get-contexts my-context
```

6.1.21. oc config get-users

显示 kubeconfig 中定义的用户

用法示例

```
# List the users that oc knows about
oc config get-users
```

6.1.22. oc config new-admin-kubeconfig

生成，使服务器信任并显示新的 admin.kubeconfig。

用法示例

```
# Generate a new admin kubeconfig
oc config new-admin-kubeconfig
```

6.1.23. oc config new-kubelet-bootstrap-kubeconfig

生成，使服务器信任并显示新的 kubelet /etc/kubernetes/kubeconfig。

用法示例

```
# Generate a new kubelet bootstrap kubeconfig
oc config new-kubelet-bootstrap-kubeconfig
```

6.1.24. oc config refresh-ca-bundle

通过联系 apiserver 来更新 OpenShift CA 捆绑包。

用法示例

```
# Refresh the CA bundle for the current context's cluster
oc config refresh-ca-bundle

# Refresh the CA bundle for the cluster named e2e in your kubeconfig
oc config refresh-ca-bundle e2e

# Print the CA bundle from the current OpenShift cluster's apiserver.
oc config refresh-ca-bundle --dry-run
```

6.1.25. oc config rename-context

从 kubeconfig 文件中重命名上下文

用法示例

```
# Rename the context 'old-name' to 'new-name' in your kubeconfig file
oc config rename-context old-name new-name
```

6.1.26. oc config set

在 kubeconfig 文件中设置单个值

用法示例

```
# Set the server field on the my-cluster cluster to https://1.2.3.4
oc config set clusters.my-cluster.server https://1.2.3.4

# Set the certificate-authority-data field on the my-cluster cluster
```

```
oc config set clusters.my-cluster.certificate-authority-data $(echo "cert_data_here" | base64 -i -)

# Set the cluster field in the my-context context to my-cluster
oc config set contexts.my-context.cluster my-cluster

# Set the client-key-data field in the cluster-admin user using --set-raw-bytes option
oc config set users.cluster-admin.client-key-data cert_data_here --set-raw-bytes=true
```

6.1.27. oc config set-cluster

在 kubeconfig 中设置集群条目

用法示例

```
# Set only the server field on the e2e cluster entry without touching other values
oc config set-cluster e2e --server=https://1.2.3.4

# Embed certificate authority data for the e2e cluster entry
oc config set-cluster e2e --embed-certs --certificate-authority=~/.kube/e2e/kubernetes.ca.crt

# Disable cert checking for the e2e cluster entry
oc config set-cluster e2e --insecure-skip-tls-verify=true

# Set custom TLS server name to use for validation for the e2e cluster entry
oc config set-cluster e2e --tls-server-name=my-cluster-name

# Set proxy url for the e2e cluster entry
oc config set-cluster e2e --proxy-url=https://1.2.3.4
```

6.1.28. oc config set-context

在 kubeconfig 中设置上下文条目

用法示例

```
# Set the user field on the gce context entry without touching other values
oc config set-context gce --user=cluster-admin
```

6.1.29. oc config set-credentials

在 kubeconfig 中设置用户条目

用法示例

```
# Set only the "client-key" field on the "cluster-admin"
# entry, without touching other values
oc config set-credentials cluster-admin --client-key=~/.kube/admin.key

# Set basic auth for the "cluster-admin" entry
oc config set-credentials cluster-admin --username=admin --password=uXFGweU9l35qcif

# Embed client certificate data in the "cluster-admin" entry
oc config set-credentials cluster-admin --client-certificate=~/.kube/admin.crt --embed-certs=true
```

```
  # Enable the Google Compute Platform auth provider for the "cluster-admin" entry
  oc config set-credentials cluster-admin --auth-provider=gcp

  # Enable the OpenID Connect auth provider for the "cluster-admin" entry with additional args
  oc config set-credentials cluster-admin --auth-provider=oidc --auth-provider-arg=client-id=foo --auth-provider-arg=client-secret=bar

  # Remove the "client-secret" config value for the OpenID Connect auth provider for the "cluster-admin" entry
  oc config set-credentials cluster-admin --auth-provider=oidc --auth-provider-arg=client-secret-

  # Enable new exec auth plugin for the "cluster-admin" entry
  oc config set-credentials cluster-admin --exec-command=/path/to/the/executable --exec-api-version=client.authentication.k8s.io/v1beta1

  # Define new exec auth plugin args for the "cluster-admin" entry
  oc config set-credentials cluster-admin --exec-arg=arg1 --exec-arg=arg2

  # Create or update exec auth plugin environment variables for the "cluster-admin" entry
  oc config set-credentials cluster-admin --exec-env=key1=val1 --exec-env=key2=val2

  # Remove exec auth plugin environment variables for the "cluster-admin" entry
  oc config set-credentials cluster-admin --exec-env=var-to-remove-
```

6.1.30. oc config unset

在 kubeconfig 文件中取消设置单个值

用法示例

```
  # Unset the current-context
  oc config unset current-context

  # Unset namespace in foo context
  oc config unset contexts.foo.namespace
```

6.1.31. oc config use-context

在 kubeconfig 文件中设置 current-context

用法示例

```
  # Use the context for the minikube cluster
  oc config use-context minikube
```

6.1.32. oc config view

显示合并的 kubeconfig 设置或指定的 kubeconfig 文件

用法示例

```
  # Show merged kubeconfig settings
```

```
oc config view

# Show merged kubeconfig settings and raw certificate data and exposed secrets
oc config view --raw

# Get the password for the e2e user
oc config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'
```

6.1.33. oc cp

将文件和目录复制到容器或从容器中复制

用法示例

```
# !!!Important Note!!!
# Requires that the 'tar' binary is present in your container
# image.  If 'tar' is not present, 'oc cp' will fail.
#
# For advanced use cases, such as symlinks, wildcard expansion or
# file mode preservation, consider using 'oc exec'.

# Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace <some-namespace>
tar cf - /tmp/foo | oc exec -i -n <some-namespace> <some-pod> -- tar xf - -C /tmp/bar

# Copy /tmp/foo from a remote pod to /tmp/bar locally
oc exec -n <some-namespace> <some-pod> -- tar cf - /tmp/foo | tar xf - -C /tmp/bar

# Copy /tmp/foo_dir local directory to /tmp/bar_dir in a remote pod in the default namespace
oc cp /tmp/foo_dir <some-pod>:/tmp/bar_dir

# Copy /tmp/foo local file to /tmp/bar in a remote pod in a specific container
oc cp /tmp/foo <some-pod>:/tmp/bar -c <specific-container>

# Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace <some-namespace>
oc cp /tmp/foo <some-namespace>/<some-pod>:/tmp/bar

# Copy /tmp/foo from a remote pod to /tmp/bar locally
oc cp <some-namespace>/<some-pod>:/tmp/foo /tmp/bar
```

6.1.34. oc create

从文件或 stdin 创建资源

用法示例

```
# Create a pod using the data in pod.json
oc create -f ./pod.json

# Create a pod based on the JSON passed into stdin
cat pod.json | oc create -f -

# Edit the data in registry.yaml in JSON then create the resource using the edited data
oc create -f registry.yaml --edit -o json
```

6.1.35. oc create clusterrole

创建集群角色

用法示例

```
  # Create a cluster role named "pod-reader" that allows user to perform "get", "watch" and "list" on pods
  oc create clusterrole pod-reader --verb=get,list,watch --resource=pods

  # Create a cluster role named "pod-reader" with ResourceName specified
  oc create clusterrole pod-reader --verb=get --resource=pods --resource-name=readablepod --resource-name=anotherpod

  # Create a cluster role named "foo" with API Group specified
  oc create clusterrole foo --verb=get,list,watch --resource=rs.apps

  # Create a cluster role named "foo" with SubResource specified
  oc create clusterrole foo --verb=get,list,watch --resource=pods,pods/status

  # Create a cluster role name "foo" with NonResourceURL specified
  oc create clusterrole "foo" --verb=get --non-resource-url=/logs/*

  # Create a cluster role name "monitoring" with AggregationRule specified
  oc create clusterrole monitoring --aggregation-rule="rbac.example.com/aggregate-to-monitoring=true"
```

6.1.36. oc create clusterrolebinding

为特定集群角色创建集群角色绑定

用法示例

```
  # Create a cluster role binding for user1, user2, and group1 using the cluster-admin cluster role
  oc create clusterrolebinding cluster-admin --clusterrole=cluster-admin --user=user1 --user=user2 --group=group1
```

6.1.37. oc create configmap

从本地文件、目录或字面值创建配置映射

用法示例

```
  # Create a new config map named my-config based on folder bar
  oc create configmap my-config --from-file=path/to/bar

  # Create a new config map named my-config with specified keys instead of file basenames on disk
  oc create configmap my-config --from-file=key1=/path/to/bar/file1.txt --from-file=key2=/path/to/bar/file2.txt

  # Create a new config map named my-config with key1=config1 and key2=config2
  oc create configmap my-config --from-literal=key1=config1 --from-literal=key2=config2

  # Create a new config map named my-config from the key=value pairs in the file
```

```
oc create configmap my-config --from-file=path/to/bar

# Create a new config map named my-config from an env file
oc create configmap my-config --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

6.1.38. oc create cronjob

使用指定名称创建 cron 作业

用法示例

```
# Create a cron job
oc create cronjob my-job --image=busybox --schedule="*/1 * * * *"

# Create a cron job with a command
oc create cronjob my-job --image=busybox --schedule="*/1 * * * *" -- date
```

6.1.39. oc create deployment

使用指定名称创建部署

用法示例

```
# Create a deployment named my-dep that runs the busybox image
oc create deployment my-dep --image=busybox

# Create a deployment with a command
oc create deployment my-dep --image=busybox -- date

# Create a deployment named my-dep that runs the nginx image with 3 replicas
oc create deployment my-dep --image=nginx --replicas=3

# Create a deployment named my-dep that runs the busybox image and expose port 5701
oc create deployment my-dep --image=busybox --port=5701
```

6.1.40. oc create ingress

使用指定名称创建入口

用法示例

```
# Create a single ingress called 'simple' that directs requests to foo.com/bar to svc
# svc1:8080 with a tls secret "my-cert"
oc create ingress simple --rule="foo.com/bar=svc1:8080,tls=my-cert"

# Create a catch all ingress of "/path" pointing to service svc:port and Ingress Class as
"otheringress"
oc create ingress catch-all --class=otheringress --rule="/path=svc:port"

# Create an ingress with two annotations: ingress.annotation1 and ingress.annotations2
oc create ingress annotated --class=default --rule="foo.com/bar=svc:port" \
--annotation ingress.annotation1=foo \
--annotation ingress.annotation2=bla
```

```
# Create an ingress with the same host and multiple paths
oc create ingress multipath --class=default \
--rule="foo.com/=svc:port" \
--rule="foo.com/admin/=svcadmin:portadmin"

# Create an ingress with multiple hosts and the pathType as Prefix
oc create ingress ingress1 --class=default \
--rule="foo.com/path*=svc:8080" \
--rule="bar.com/admin*=svc2:http"

# Create an ingress with TLS enabled using the default ingress certificate and different path types
oc create ingress ingtls --class=default \
--rule="foo.com/=svc:https,tls" \
--rule="foo.com/path/subpath*=othersvc:8080"

# Create an ingress with TLS enabled using a specific secret and pathType as Prefix
oc create ingress ingsecret --class=default \
--rule="foo.com/*=svc:8080,tls=secret1"

# Create an ingress with a default backend
oc create ingress ingdefault --class=default \
--default-backend=defaultsvc:http \
--rule="foo.com/*=svc:8080,tls=secret1"
```

6.1.41. oc create job

使用指定名称创建作业

用法示例

```
# Create a job
oc create job my-job --image=busybox

# Create a job with a command
oc create job my-job --image=busybox -- date

# Create a job from a cron job named "a-cronjob"
oc create job test-job --from=cronjob/a-cronjob
```

6.1.42. oc create namespace

使用指定名称创建命名空间

用法示例

```
# Create a new namespace named my-namespace
oc create namespace my-namespace
```

6.1.43. oc create poddisruptionBudget

使用指定名称创建 pod 中断预算

用法示例

```
# Create a pod disruption budget named my-pdb that will select all pods with the app=rails label
# and require at least one of them being available at any point in time
oc create poddisruptionbudget my-pdb --selector=app=rails --min-available=1

# Create a pod disruption budget named my-pdb that will select all pods with the app=nginx label
# and require at least half of the pods selected to be available at any point in time
oc create pdb my-pdb --selector=app=nginx --min-available=50%
```

6.1.44. oc create priorityclass

创建具有指定名称的优先级类

用法示例

```
# Create a priority class named high-priority
oc create priorityclass high-priority --value=1000 --description="high priority"

# Create a priority class named default-priority that is considered as the global default priority
oc create priorityclass default-priority --value=1000 --global-default=true --description="default
priority"

# Create a priority class named high-priority that cannot preempt pods with lower priority
oc create priorityclass high-priority --value=1000 --description="high priority" --preemption-
policy="Never"
```

6.1.45. oc create quota

使用指定名称创建配额

用法示例

```
# Create a new resource quota named my-quota
oc create quota my-quota --
hard=cpu=1,memory=1G,pods=2,services=3,replicationcontrollers=2,resourcequotas=1,secrets=5,persis
tentvolumeclaims=10

# Create a new resource quota named best-effort
oc create quota best-effort --hard=pods=100 --scopes=BestEffort
```

6.1.46. oc create role

创建具有单一规则的角色

用法示例

```
# Create a role named "pod-reader" that allows user to perform "get", "watch" and "list" on pods
oc create role pod-reader --verb=get --verb=list --verb=watch --resource=pods

# Create a role named "pod-reader" with ResourceName specified
oc create role pod-reader --verb=get --resource=pods --resource-name=readablepod --resource-
name=anotherpod
```

```
# Create a role named "foo" with API Group specified
oc create role foo --verb=get,list,watch --resource=rs.apps

# Create a role named "foo" with SubResource specified
oc create role foo --verb=get,list,watch --resource=pods,pods/status
```

6.1.47. oc create rolebinding

为特定角色或集群角色创建角色绑定

用法示例

```
# Create a role binding for user1, user2, and group1 using the admin cluster role
oc create rolebinding admin --clusterrole=admin --user=user1 --user=user2 --group=group1

# Create a role binding for serviceaccount monitoring:sa-dev using the admin role
oc create rolebinding admin-binding --role=admin --serviceaccount=monitoring:sa-dev
```

6.1.48. oc create route edge

创建使用边缘 TLS 终止的路由

用法示例

```
# Create an edge route named "my-route" that exposes the frontend service
oc create route edge my-route --service=frontend

# Create an edge route that exposes the frontend service and specify a path
# If the route name is omitted, the service name will be used
oc create route edge --service=frontend --path /assets
```

6.1.49. oc create route passthrough

创建使用 passthrough TLS 终止的路由

用法示例

```
# Create a passthrough route named "my-route" that exposes the frontend service
oc create route passthrough my-route --service=frontend

# Create a passthrough route that exposes the frontend service and specify
# a host name. If the route name is omitted, the service name will be used
oc create route passthrough --service=frontend --hostname=www.example.com
```

6.1.50. oc create route reencrypt

创建使用重新加密 TLS 终止的路由

用法示例

```
# Create a route named "my-route" that exposes the frontend service
```

```
  oc create route reencrypt my-route --service=frontend --dest-ca-cert cert.cert

  # Create a reencrypt route that exposes the frontend service, letting the
  # route name default to the service name and the destination CA certificate
  # default to the service CA
  oc create route reencrypt --service=frontend
```

6.1.51. oc create secret docker-registry

创建用于 Docker registry 的 secret

用法示例

```
  # If you don't already have a .dockercfg file, you can create a dockercfg secret directly by using:
  oc create secret docker-registry my-secret --docker-server=DOCKER_REGISTRY_SERVER --
docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-
email=DOCKER_EMAIL

  # Create a new secret named my-secret from ~/.docker/config.json
  oc create secret docker-registry my-secret --from-file=.dockerconfigjson=path/to/.docker/config.json
```

6.1.52. oc create secret generic

从本地文件、目录或字面值创建 secret

用法示例

```
  # Create a new secret named my-secret with keys for each file in folder bar
  oc create secret generic my-secret --from-file=path/to/bar

  # Create a new secret named my-secret with specified keys instead of names on disk
  oc create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-file=ssh-
publickey=path/to/id_rsa.pub

  # Create a new secret named my-secret with key1=supersecret and key2=topsecret
  oc create secret generic my-secret --from-literal=key1=supersecret --from-literal=key2=topsecret

  # Create a new secret named my-secret using a combination of a file and a literal
  oc create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-
literal=passphrase=topsecret

  # Create a new secret named my-secret from env files
  oc create secret generic my-secret --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

6.1.53. oc create secret tls

创建 TLS secret

用法示例

```
  # Create a new TLS secret named tls-secret with the given key pair
  oc create secret tls tls-secret --cert=path/to/tls.cert --key=path/to/tls.key
```

6.1.54. oc create service clusterip

创建 ClusterIP 服务

用法示例

```
# Create a new ClusterIP service named my-cs
oc create service clusterip my-cs --tcp=5678:8080

# Create a new ClusterIP service named my-cs (in headless mode)
oc create service clusterip my-cs --clusterip="None"
```

6.1.55. oc create service externalname

创建 ExternalName 服务

用法示例

```
# Create a new ExternalName service named my-ns
oc create service externalname my-ns --external-name bar.com
```

6.1.56. oc create service loadbalancer

创建 LoadBalancer 服务

用法示例

```
# Create a new LoadBalancer service named my-lbs
oc create service loadbalancer my-lbs --tcp=5678:8080
```

6.1.57. oc create service nodeport

创建 NodePort 服务

用法示例

```
# Create a new NodePort service named my-ns
oc create service nodeport my-ns --tcp=5678:8080
```

6.1.58. oc create serviceaccount

使用指定名称创建服务帐户

用法示例

```
# Create a new service account named my-service-account
oc create serviceaccount my-service-account
```

6.1.59. oc create token

请求服务帐户令牌

用法示例

```
  # Request a token to authenticate to the kube-apiserver as the service account "myapp" in the
current namespace
  oc create token myapp

  # Request a token for a service account in a custom namespace
  oc create token myapp --namespace myns

  # Request a token with a custom expiration
  oc create token myapp --duration 10m

  # Request a token with a custom audience
  oc create token myapp --audience https://example.com

  # Request a token bound to an instance of a Secret object
  oc create token myapp --bound-object-kind Secret --bound-object-name mysecret

  # Request a token bound to an instance of a Secret object with a specific uid
  oc create token myapp --bound-object-kind Secret --bound-object-name mysecret --bound-object-
uid 0d4691ed-659b-4935-a832-355f77ee47cc
```

6.1.60. oc debug

启动用于调试的 pod 的新实例

用法示例

```
  # Start a shell session into a pod using the OpenShift tools image
  oc debug

  # Debug a currently running deployment by creating a new pod
  oc debug deploy/test

  # Debug a node as an administrator
  oc debug node/master-1

  # Launch a shell in a pod using the provided image stream tag
  oc debug istag/mysql:latest -n openshift

  # Test running a job as a non-root user
  oc debug job/test --as-user=1000000

  # Debug a specific failing container by running the env command in the 'second' container
  oc debug daemonset/test -c second -- /bin/env

  # See the pod that would be created to debug
  oc debug mypod-9xbc -o yaml

  # Debug a resource but launch the debug pod in another namespace
  # Note: Not all resources can be debugged using --to-namespace without modification. For
example,
  # volumes and service accounts are namespace-dependent. Add '-o yaml' to output the debug pod
```

```
definition
  # to disk.  If necessary, edit the definition then run 'oc debug -f -' or run without --to-namespace
  oc debug mypod-9xbc --to-namespace testns
```

6.1.61. oc delete

通过文件名、stdin、资源和名称或者资源和标签选择器删除资源

用法示例

```
  # Delete a pod using the type and name specified in pod.json
  oc delete -f ./pod.json

  # Delete resources from a directory containing kustomization.yaml - e.g. dir/kustomization.yaml
  oc delete -k dir

  # Delete resources from all files that end with '.json' - i.e. expand wildcard characters in file names
  oc delete -f '*.json'

  # Delete a pod based on the type and name in the JSON passed into stdin
  cat pod.json | oc delete -f -

  # Delete pods and services with same names "baz" and "foo"
  oc delete pod,service baz foo

  # Delete pods and services with label name=myLabel
  oc delete pods,services -l name=myLabel

  # Delete a pod with minimal delay
  oc delete pod foo --now

  # Force delete a pod on a dead node
  oc delete pod foo --force

  # Delete all pods
  oc delete pods --all
```

6.1.62. oc describe

显示特定资源或一组资源的详情

用法示例

```
  # Describe a node
  oc describe nodes kubernetes-node-emt8.c.myproject.internal

  # Describe a pod
  oc describe pods/nginx

  # Describe a pod identified by type and name in "pod.json"
  oc describe -f pod.json

  # Describe all pods
  oc describe pods
```

```
# Describe pods by label name=myLabel
oc describe po -l name=myLabel

# Describe all pods managed by the 'frontend' replication controller
# (rc-created pods get the name of the rc as a prefix in the pod name)
oc describe pods frontend
```

6.1.63. oc diff

针对 would-be 应用的版本对 live 版本进行 diff 操作

用法示例

```
# Diff resources included in pod.json
oc diff -f pod.json

# Diff file read from stdin
cat service.yaml | oc diff -f -
```

6.1.64. oc edit

编辑服务器上的资源

用法示例

```
# Edit the service named 'registry'
oc edit svc/registry

# Use an alternative editor
KUBE_EDITOR="nano" oc edit svc/registry

# Edit the job 'myjob' in JSON using the v1 API format
oc edit job.v1.batch/myjob -o json

# Edit the deployment 'mydeployment' in YAML and save the modified config in its annotation
oc edit deployment/mydeployment -o yaml --save-config

# Edit the deployment/mydeployment's status subresource
oc edit deployment mydeployment --subresource='status'
```

6.1.65. oc 事件

列出事件

用法示例

```
# List recent events in the default namespace.
oc events

# List recent events in all namespaces.
oc events --all-namespaces
```

```
# List recent events for the specified pod, then wait for more events and list them as they arrive.
oc events --for pod/web-pod-13je7 --watch

# List recent events in given format. Supported ones, apart from default, are json and yaml.
oc events -oyaml

# List recent only events in given event types
oc events --types=Warning,Normal
```

6.1.66. oc exec

在容器中执行命令

用法示例

```
# Get output from running the 'date' command from pod mypod, using the first container by default
oc exec mypod -- date

# Get output from running the 'date' command in ruby-container from pod mypod
oc exec mypod -c ruby-container -- date

# Switch to raw terminal mode; sends stdin to 'bash' in ruby-container from pod mypod
# and sends stdout/stderr from 'bash' back to the client
oc exec mypod -c ruby-container -i -t -- bash -il

# List contents of /usr from the first container of pod mypod and sort by modification time
# If the command you want to execute in the pod has any flags in common (e.g. -i),
# you must use two dashes (--) to separate your command's flags/arguments
# Also note, do not surround your command and its flags/arguments with quotes
# unless that is how you would execute it normally (i.e., do ls -t /usr, not "ls -t /usr")
oc exec mypod -i -t -- ls -t /usr

# Get output from running 'date' command from the first pod of the deployment mydeployment,
using the first container by default
oc exec deploy/mydeployment -- date

# Get output from running 'date' command from the first pod of the service myservice, using the first
container by default
oc exec svc/myservice -- date
```

6.1.67. oc explain

获取资源的文档

用法示例

```
# Get the documentation of the resource and its fields
oc explain pods

# Get the documentation of a specific field of a resource
oc explain pods.spec.containers
```

6.1.68. oc expose

将复制的应用程序作为服务或路由公开

用法示例

```
# Create a route based on service nginx. The new route will reuse nginx's labels
oc expose service nginx

# Create a route and specify your own label and route name
oc expose service nginx -l name=myroute --name=fromdowntown

# Create a route and specify a host name
oc expose service nginx --hostname=www.example.com

# Create a route with a wildcard
oc expose service nginx --hostname=x.example.com --wildcard-policy=Subdomain
# This would be equivalent to *.example.com. NOTE: only hosts are matched by the wildcard;
subdomains would not be included

# Expose a deployment configuration as a service and use the specified port
oc expose dc ruby-hello-world --port=8080

# Expose a service as a route in the specified path
oc expose service nginx --path=/nginx
```

6.1.69. oc extract

将 secret 或配置映射提取到磁盘

用法示例

```
# Extract the secret "test" to the current directory
oc extract secret/test

# Extract the config map "nginx" to the /tmp directory
oc extract configmap/nginx --to=/tmp

# Extract the config map "nginx" to STDOUT
oc extract configmap/nginx --to=-

# Extract only the key "nginx.conf" from config map "nginx" to the /tmp directory
oc extract configmap/nginx --to=/tmp --keys=nginx.conf
```

6.1.70. oc get

显示一个或多个资源

用法示例

```
# List all pods in ps output format
oc get pods

# List all pods in ps output format with more information (such as node name)
oc get pods -o wide
```

```
# List a single replication controller with specified NAME in ps output format
oc get replicationcontroller web

# List deployments in JSON output format, in the "v1" version of the "apps" API group
oc get deployments.v1.apps -o json

# List a single pod in JSON output format
oc get -o json pod web-pod-13je7

# List a pod identified by type and name specified in "pod.yaml" in JSON output format
oc get -f pod.yaml -o json

# List resources from a directory with kustomization.yaml - e.g. dir/kustomization.yaml
oc get -k dir/

# Return only the phase value of the specified pod
oc get -o template pod/web-pod-13je7 --template={{.status.phase}}

# List resource information in custom columns
oc get pod test-pod -o custom-
columns=CONTAINER:.spec.containers[0].name,IMAGE:.spec.containers[0].image

# List all replication controllers and services together in ps output format
oc get rc,services

# List one or more resources by their type and names
oc get rc/web service/frontend pods/web-pod-13je7

# List status subresource for a single pod.
oc get pod web-pod-13je7 --subresource status
```

6.1.71. oc image append

向镜像添加层并将其推送到 registry

用法示例

```
# Remove the entrypoint on the mysql:latest image
oc image append --from mysql:latest --to myregistry.com/myimage:latest --image '{"Entrypoint":null}'

# Add a new layer to the image
oc image append --from mysql:latest --to myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to the image and store the result on disk
# This results in $(pwd)/v2/mysql/blobs,manifests
oc image append --from mysql:latest --to file://mysql:local layer.tar.gz

# Add a new layer to the image and store the result on disk in a designated directory
# This will result in $(pwd)/mysql-local/v2/mysql/blobs,manifests
oc image append --from mysql:latest --to file://mysql:local --dir mysql-local layer.tar.gz

# Add a new layer to an image that is stored on disk (~/mysql-local/v2/image exists)
oc image append --from-dir ~/mysql-local --to myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to an image that was mirrored to the current directory on disk ($(pwd)/v2/image
```

```
exists)
  oc image append --from-dir v2 --to myregistry.com/myimage:latest layer.tar.gz

  # Add a new layer to a multi-architecture image for an os/arch that is different from the system's
os/arch
  # Note: The first image in the manifest list that matches the filter will be returned when --keep-
manifest-list is specified
  oc image append --from docker.io/library/busybox:latest --filter-by-os=linux/s390x --to
myregistry.com/myimage:latest layer.tar.gz

  # Add a new layer to a multi-architecture image for all the os/arch manifests when keep-manifest-list
is specified
  oc image append --from docker.io/library/busybox:latest --keep-manifest-list --to
myregistry.com/myimage:latest layer.tar.gz

  # Add a new layer to a multi-architecture image for all the os/arch manifests that is specified by the
filter, while preserving the manifestlist
  oc image append --from docker.io/library/busybox:latest --filter-by-os=linux/s390x --keep-manifest-
list --to myregistry.com/myimage:latest layer.tar.gz
```

6.1.72. oc image extract

将文件从镜像复制到文件系统

用法示例

```
  # Extract the busybox image into the current directory
  oc image extract docker.io/library/busybox:latest

  # Extract the busybox image into a designated directory (must exist)
  oc image extract docker.io/library/busybox:latest --path /:/tmp/busybox

  # Extract the busybox image into the current directory for linux/s390x platform
  # Note: Wildcard filter is not supported with extract; pass a single os/arch to extract
  oc image extract docker.io/library/busybox:latest --filter-by-os=linux/s390x

  # Extract a single file from the image into the current directory
  oc image extract docker.io/library/centos:7 --path /bin/bash:.

  # Extract all .repo files from the image's /etc/yum.repos.d/ folder into the current directory
  oc image extract docker.io/library/centos:7 --path /etc/yum.repos.d/*.repo:.

  # Extract all .repo files from the image's /etc/yum.repos.d/ folder into a designated directory (must
exist)
  # This results in /tmp/yum.repos.d/*.repo on local system
  oc image extract docker.io/library/centos:7 --path /etc/yum.repos.d/*.repo:/tmp/yum.repos.d

  # Extract an image stored on disk into the current directory ($(pwd)/v2/busybox/blobs,manifests
exists)
  # --confirm is required because the current directory is not empty
  oc image extract file://busybox:local --confirm

  # Extract an image stored on disk in a directory other than $(pwd)/v2 into the current directory
  # --confirm is required because the current directory is not empty ($(pwd)/busybox-mirror-
dir/v2/busybox exists)
```

```
  oc image extract file://busybox:local --dir busybox-mirror-dir --confirm

  # Extract an image stored on disk in a directory other than $(pwd)/v2 into a designated directory
(must exist)
  oc image extract file://busybox:local --dir busybox-mirror-dir --path /:/tmp/busybox

  # Extract the last layer in the image
  oc image extract docker.io/library/centos:7[-1]

  # Extract the first three layers of the image
  oc image extract docker.io/library/centos:7[:3]

  # Extract the last three layers of the image
  oc image extract docker.io/library/centos:7[-3:]
```

6.1.73. oc image info

显示镜像的信息

用法示例

```
  # Show information about an image
  oc image info quay.io/openshift/cli:latest

  # Show information about images matching a wildcard
  oc image info quay.io/openshift/cli:4.*

  # Show information about a file mirrored to disk under DIR
  oc image info --dir=DIR file://library/busybox:latest

  # Select which image from a multi-OS image to show
  oc image info library/busybox:latest --filter-by-os=linux/arm64
```

6.1.74. oc image mirror

将镜像从一个存储库镜像到另一个存储库

用法示例

```
  # Copy image to another tag
  oc image mirror myregistry.com/myimage:latest myregistry.com/myimage:stable

  # Copy image to another registry
  oc image mirror myregistry.com/myimage:latest docker.io/myrepository/myimage:stable

  # Copy all tags starting with mysql to the destination repository
  oc image mirror myregistry.com/myimage:mysql* docker.io/myrepository/myimage

  # Copy image to disk, creating a directory structure that can be served as a registry
  oc image mirror myregistry.com/myimage:latest file://myrepository/myimage:latest

  # Copy image to S3 (pull from <bucket>.s3.amazonaws.com/image:latest)
  oc image mirror myregistry.com/myimage:latest
s3://s3.amazonaws.com/<region>/<bucket>/image:latest
```

```
  # Copy image to S3 without setting a tag (pull via @<digest>)
  oc image mirror myregistry.com/myimage:latest s3://s3.amazonaws.com/<region>/<bucket>/image

  # Copy image to multiple locations
  oc image mirror myregistry.com/myimage:latest docker.io/myrepository/myimage:stable \
  docker.io/myrepository/myimage:dev

  # Copy multiple images
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  myregistry.com/myimage:new=myregistry.com/other:target

  # Copy manifest list of a multi-architecture image, even if only a single image is found
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --keep-manifest-list=true

  # Copy specific os/arch manifest of a multi-architecture image
  # Run 'oc image info myregistry.com/myimage:latest' to see available os/arch for multi-arch images
  # Note that with multi-arch images, this results in a new manifest list digest that includes only
  # the filtered manifests
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --filter-by-os=os/arch

  # Copy all os/arch manifests of a multi-architecture image
  # Run 'oc image info myregistry.com/myimage:latest' to see list of os/arch manifests that will be
mirrored
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --keep-manifest-list=true

  # Note the above command is equivalent to
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --filter-by-os=.*

  # Copy specific os/arch manifest of a multi-architecture image
  # Run 'oc image info myregistry.com/myimage:latest' to see available os/arch for multi-arch images
  # Note that the target registry may reject a manifest list if the platform specific images do not all
  # exist. You must use a registry with sparse registry support enabled.
  oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
  --filter-by-os=os/arch \
  --keep-manifest-list=true
```

6.1.75. oc kustomize

从目录或 URL 构建 kustomization 目标

用法示例

```
  # Build the current working directory
  oc kustomize

  # Build some shared configuration directory
  oc kustomize /home/config/production

  # Build from github
  oc kustomize https://github.com/kubernetes-sigs/kustomize.git/examples/helloWorld?ref=v1.0.6
```

6.1.76. oc label

更新资源上的标签

用法示例

```
# Update pod 'foo' with the label 'unhealthy' and the value 'true'
oc label pods foo unhealthy=true

# Update pod 'foo' with the label 'status' and the value 'unhealthy', overwriting any existing value
oc label --overwrite pods foo status=unhealthy

# Update all pods in the namespace
oc label pods --all status=unhealthy

# Update a pod identified by the type and name in "pod.json"
oc label -f pod.json status=unhealthy

# Update pod 'foo' only if the resource is unchanged from version 1
oc label pods foo status=unhealthy --resource-version=1

# Update pod 'foo' by removing a label named 'bar' if it exists
# Does not require the --overwrite flag
oc label pods foo bar-
```

6.1.77. oc logs

显示 pod 中容器的日志

用法示例

```
# Start streaming the logs of the most recent build of the openldap build config
oc logs -f bc/openldap

# Start streaming the logs of the latest deployment of the mysql deployment config
oc logs -f dc/mysql

# Get the logs of the first deployment for the mysql deployment config. Note that logs
# from older deployments may not exist either because the deployment was successful
# or due to deployment pruning or manual deletion of the deployment
oc logs --version=1 dc/mysql

# Return a snapshot of ruby-container logs from pod backend
oc logs backend -c ruby-container

# Start streaming of ruby-container logs from pod backend
oc logs -f pod/backend -c ruby-container
```

6.1.78. oc observe

观察资源的变化并对其做出反应（实验性）

用法示例

```
# Observe changes to services
oc observe services

# Observe changes to services, including the clusterIP and invoke a script for each
oc observe services --template '{ .spec.clusterIP }' -- register_dns.sh

# Observe changes to services filtered by a label selector
oc observe services -l regist-dns=true --template '{ .spec.clusterIP }' -- register_dns.sh
```

6.1.79. oc patch

更新资源字段

用法示例

```
# Partially update a node using a strategic merge patch, specifying the patch as JSON
oc patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}'

# Partially update a node using a strategic merge patch, specifying the patch as YAML
oc patch node k8s-node-1 -p $'spec:\n unschedulable: true'

# Partially update a node identified by the type and name specified in "node.json" using strategic merge patch
oc patch -f node.json -p '{"spec":{"unschedulable":true}}'

# Update a container's image; spec.containers[*].name is required because it's a merge key
oc patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serve-hostname","image":"new image"}]}}'

# Update a container's image using a JSON patch with positional arrays
oc patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spec/containers/0/image", "value":"new image"}]'

# Update a deployment's replicas through the scale subresource using a merge patch.
oc patch deployment nginx-deployment --subresource='scale' --type='merge' -p '{"spec":{"replicas":2}}'
```

6.1.80. oc plugin list

列出用户 PATH 中的所有可见插件可执行文件

用法示例

```
# List all available plugins
oc plugin list
```

6.1.81. oc policy add-role-to-user

为当前项目的用户或服务帐户添加角色

用法示例

```
# Add the 'view' role to user1 for the current project
```

```
oc policy add-role-to-user view user1

# Add the 'edit' role to serviceaccount1 for the current project
oc policy add-role-to-user edit -z serviceaccount1
```

6.1.82. oc policy scc-review

检查哪个服务帐户可以创建 pod

用法示例

```
# Check whether service accounts sa1 and sa2 can admit a pod with a template pod spec specified in my_resource.yaml
# Service Account specified in myresource.yaml file is ignored
oc policy scc-review -z sa1,sa2 -f my_resource.yaml

# Check whether service accounts system:serviceaccount:bob:default can admit a pod with a template pod spec specified in my_resource.yaml
oc policy scc-review -z system:serviceaccount:bob:default -f my_resource.yaml

# Check whether the service account specified in my_resource_with_sa.yaml can admit the pod
oc policy scc-review -f my_resource_with_sa.yaml

# Check whether the default service account can admit the pod; default is taken since no service account is defined in myresource_with_no_sa.yaml
oc policy scc-review -f myresource_with_no_sa.yaml
```

6.1.83. oc policy scc-subject-review

检查用户或服务帐户是否可以创建 pod

用法示例

```
# Check whether user bob can create a pod specified in myresource.yaml
oc policy scc-subject-review -u bob -f myresource.yaml

# Check whether user bob who belongs to projectAdmin group can create a pod specified in myresource.yaml
oc policy scc-subject-review -u bob -g projectAdmin -f myresource.yaml

# Check whether a service account specified in the pod template spec in myresourcewithsa.yaml can create the pod
oc policy scc-subject-review -f myresourcewithsa.yaml
```

6.1.84. oc port-forward

将一个或多个本地端口转发到一个 pod

用法示例

```
# Listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in the pod
oc port-forward pod/mypod 5000 6000
```

```
  # Listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in a pod selected by the deployment
  oc port-forward deployment/mydeployment 5000 6000

  # Listen on port 8443 locally, forwarding to the targetPort of the service's port named "https" in a pod selected by the service
  oc port-forward service/myservice 8443:https

  # Listen on port 8888 locally, forwarding to 5000 in the pod
  oc port-forward pod/mypod 8888:5000

  # Listen on port 8888 on all addresses, forwarding to 5000 in the pod
  oc port-forward --address 0.0.0.0 pod/mypod 8888:5000

  # Listen on port 8888 on localhost and selected IP, forwarding to 5000 in the pod
  oc port-forward --address localhost,10.19.21.23 pod/mypod 8888:5000

  # Listen on a random port locally, forwarding to 5000 in the pod
  oc port-forward pod/mypod :5000
```

6.1.85. oc proxy

运行到 Kubernetes API 服务器的代理

用法示例

```
  # To proxy all of the Kubernetes API and nothing else
  oc proxy --api-prefix=/

  # To proxy only part of the Kubernetes API and also some static files
  # You can get pods info with 'curl localhost:8001/api/v1/pods'
  oc proxy --www=/my/files --www-prefix=/static/ --api-prefix=/api/

  # To proxy the entire Kubernetes API at a different root
  # You can get pods info with 'curl localhost:8001/custom/api/v1/pods'
  oc proxy --api-prefix=/custom/

  # Run a proxy to the Kubernetes API server on port 8011, serving static content from ./local/www/
  oc proxy --port=8011 --www=./local/www/

  # Run a proxy to the Kubernetes API server on an arbitrary local port
  # The chosen port for the server will be output to stdout
  oc proxy --port=0

  # Run a proxy to the Kubernetes API server, changing the API prefix to k8s-api
  # This makes e.g. the pods API available at localhost:8001/k8s-api/v1/pods/
  oc proxy --api-prefix=/k8s-api
```

6.1.86. oc rollback

将应用程序的一部分还原回以前的部署

用法示例

```
# Perform a rollback to the last successfully completed deployment for a deployment config
oc rollback frontend

# See what a rollback to version 3 will look like, but do not perform the rollback
oc rollback frontend --to-version=3 --dry-run

# Perform a rollback to a specific deployment
oc rollback frontend-2

# Perform the rollback manually by piping the JSON of the new config back to oc
oc rollback frontend -o json | oc replace dc/frontend -f -

# Print the updated deployment configuration in JSON format instead of performing the rollback
oc rollback frontend -o json
```

6.1.87. oc rollout cancel

取消进行中的部署

用法示例

```
# Cancel the in-progress deployment based on 'nginx'
oc rollout cancel dc/nginx
```

6.1.88. oc rollout history

查看推出（rollout）历史记录

用法示例

```
# View the rollout history of a deployment
oc rollout history dc/nginx

# View the details of deployment revision 3
oc rollout history dc/nginx --revision=3
```

6.1.89. oc rollout latest

使用来自触发器的最新状态为部署配置启动一个新的 rollout 操作

用法示例

```
# Start a new rollout based on the latest images defined in the image change triggers
oc rollout latest dc/nginx

# Print the rolled out deployment config
oc rollout latest dc/nginx -o json
```

6.1.90. oc rollout pause

将提供的资源标记为暂停

用法示例

```
# Mark the nginx deployment as paused. Any current state of
# the deployment will continue its function, new updates to the deployment will not
# have an effect as long as the deployment is paused
oc rollout pause dc/nginx
```

6.1.91. oc rollout restart

重启资源

用法示例

```
# Restart a deployment
oc rollout restart deployment/nginx

# Restart a daemon set
oc rollout restart daemonset/abc

# Restart deployments with the app=nginx label
oc rollout restart deployment --selector=app=nginx
```

6.1.92. oc rollout resume

恢复暂停的资源

用法示例

```
# Resume an already paused deployment
oc rollout resume dc/nginx
```

6.1.93. oc rollout retry

重试最新失败的 rollout 操作

用法示例

```
# Retry the latest failed deployment based on 'frontend'
# The deployer pod and any hook pods are deleted for the latest failed deployment
oc rollout retry dc/frontend
```

6.1.94. oc rollout status

显示推出部署的状态

用法示例

```
# Watch the status of the latest rollout
oc rollout status dc/nginx
```

6.1.95. oc rollout undo

撤消之前的推出部署

用法示例

```
# Roll back to the previous deployment
oc rollout undo dc/nginx

# Roll back to deployment revision 3. The replication controller for that version must exist
oc rollout undo dc/nginx --to-revision=3
```

6.1.96. oc rsh

在容器中启动 shell 会话

用法示例

```
# Open a shell session on the first container in pod 'foo'
oc rsh foo

# Open a shell session on the first container in pod 'foo' and namespace 'bar'
# (Note that oc client specific arguments must come before the resource name and its arguments)
oc rsh -n bar foo

# Run the command 'cat /etc/resolv.conf' inside pod 'foo'
oc rsh foo cat /etc/resolv.conf

# See the configuration of your internal registry
oc rsh dc/docker-registry cat config.yml

# Open a shell session on the container named 'index' inside a pod of your job
oc rsh -c index job/scheduled
```

6.1.97. oc rsync

在本地文件系统和 pod 间复制文件

用法示例

```
# Synchronize a local directory with a pod directory
oc rsync ./local/dir/ POD:/remote/dir

# Synchronize a pod directory with a local directory
oc rsync POD:/remote/dir/ ./local/dir
```

6.1.98. oc run

在集群中运行特定镜像

用法示例

```
# Start a nginx pod
oc run nginx --image=nginx
```

```
  # Start a hazelcast pod and let the container expose port 5701
  oc run hazelcast --image=hazelcast/hazelcast --port=5701

  # Start a hazelcast pod and set environment variables "DNS_DOMAIN=cluster" and
"POD_NAMESPACE=default" in the container
  oc run hazelcast --image=hazelcast/hazelcast --env="DNS_DOMAIN=cluster" --
env="POD_NAMESPACE=default"

  # Start a hazelcast pod and set labels "app=hazelcast" and "env=prod" in the container
  oc run hazelcast --image=hazelcast/hazelcast --labels="app=hazelcast,env=prod"

  # Dry run; print the corresponding API objects without creating them
  oc run nginx --image=nginx --dry-run=client

  # Start a nginx pod, but overload the spec with a partial set of values parsed from JSON
  oc run nginx --image=nginx --overrides='{ "apiVersion": "v1", "spec": { ... } }'

  # Start a busybox pod and keep it in the foreground, don't restart it if it exits
  oc run -i -t busybox --image=busybox --restart=Never

  # Start the nginx pod using the default command, but use custom arguments (arg1 .. argN) for that
command
  oc run nginx --image=nginx -- <arg1> <arg2> ... <argN>

  # Start the nginx pod using a different command and custom arguments
  oc run nginx --image=nginx --command -- <cmd> <arg1> ... <argN>
```

6.1.99. oc scale

为部署、副本集或复制控制器设置新大小

用法示例

```
  # Scale a replica set named 'foo' to 3
  oc scale --replicas=3 rs/foo

  # Scale a resource identified by type and name specified in "foo.yaml" to 3
  oc scale --replicas=3 -f foo.yaml

  # If the deployment named mysql's current size is 2, scale mysql to 3
  oc scale --current-replicas=2 --replicas=3 deployment/mysql

  # Scale multiple replication controllers
  oc scale --replicas=5 rc/foo rc/bar rc/baz

  # Scale stateful set named 'web' to 3
  oc scale --replicas=3 statefulset/web
```

6.1.100. oc secrets link

将 secret 链接到服务帐户

用法示例

```
# Add an image pull secret to a service account to automatically use it for pulling pod images
oc secrets link serviceaccount-name pull-secret --for=pull

# Add an image pull secret to a service account to automatically use it for both pulling and pushing build images
oc secrets link builder builder-image-secret --for=pull,mount
```

6.1.101. oc secrets unlink

从服务帐户分离 secret

用法示例

```
# Unlink a secret currently associated with a service account
oc secrets unlink serviceaccount-name secret-name another-secret-name ...
```

6.1.102. oc set data

更新配置映射或 secret 中的数据

用法示例

```
# Set the 'password' key of a secret
oc set data secret/foo password=this_is_secret

# Remove the 'password' key from a secret
oc set data secret/foo password-

# Update the 'haproxy.conf' key of a config map from a file on disk
oc set data configmap/bar --from-file=../haproxy.conf

# Update a secret with the contents of a directory, one key per file
oc set data secret/foo --from-file=secret-dir
```

6.1.103. oc set env

更新 pod 模板上的环境变量

用法示例

```
# Update deployment config 'myapp' with a new environment variable
oc set env dc/myapp STORAGE_DIR=/local

# List the environment variables defined on a build config 'sample-build'
oc set env bc/sample-build --list

# List the environment variables defined on all pods
oc set env pods --all --list

# Output modified build config in YAML
oc set env bc/sample-build STORAGE_DIR=/data -o yaml

# Update all containers in all replication controllers in the project to have ENV=prod
```

```
oc set env rc --all ENV=prod

# Import environment from a secret
oc set env --from=secret/mysecret dc/myapp

# Import environment from a config map with a prefix
oc set env --from=configmap/myconfigmap --prefix=MYSQL_ dc/myapp

# Remove the environment variable ENV from container 'c1' in all deployment configs
oc set env dc --all --containers="c1" ENV-

# Remove the environment variable ENV from a deployment config definition on disk and
# update the deployment config on the server
oc set env -f dc.json ENV-

# Set some of the local shell environment into a deployment config on the server
oc set env | grep RAILS_ | oc env -e - dc/myapp
```

6.1.104. oc set image

更新 pod 模板的镜像

用法示例

```
  # Set a deployment config's nginx container image to 'nginx:1.9.1', and its busybox container image
to 'busybox'.
  oc set image dc/nginx busybox=busybox nginx=nginx:1.9.1

  # Set a deployment config's app container image to the image referenced by the imagestream tag
'openshift/ruby:2.3'.
  oc set image dc/myapp app=openshift/ruby:2.3 --source=imagestreamtag

  # Update all deployments' and rc's nginx container's image to 'nginx:1.9.1'
  oc set image deployments,rc nginx=nginx:1.9.1 --all

  # Update image of all containers of daemonset abc to 'nginx:1.9.1'
  oc set image daemonset abc *=nginx:1.9.1

  # Print result (in YAML format) of updating nginx container image from local file, without hitting the
server
  oc set image -f path/to/file.yaml nginx=nginx:1.9.1 --local -o yaml
```

6.1.105. oc set image-lookup

更改部署应用程序时镜像的解析方式

用法示例

```
  # Print all of the image streams and whether they resolve local names
  oc set image-lookup

  # Use local name lookup on image stream mysql
  oc set image-lookup mysql
```

```
# Force a deployment to use local name lookup
oc set image-lookup deploy/mysql

# Show the current status of the deployment lookup
oc set image-lookup deploy/mysql --list

# Disable local name lookup on image stream mysql
oc set image-lookup mysql --enabled=false

# Set local name lookup on all image streams
oc set image-lookup --all
```

6.1.106. oc set probe

更新 pod 模板上的探测

用法示例

```
# Clear both readiness and liveness probes off all containers
oc set probe dc/myapp --remove --readiness --liveness

# Set an exec action as a liveness probe to run 'echo ok'
oc set probe dc/myapp --liveness -- echo ok

# Set a readiness probe to try to open a TCP socket on 3306
oc set probe rc/mysql --readiness --open-tcp=3306

# Set an HTTP startup probe for port 8080 and path /healthz over HTTP on the pod IP
oc set probe dc/webapp --startup --get-url=http://:8080/healthz

# Set an HTTP readiness probe for port 8080 and path /healthz over HTTP on the pod IP
oc set probe dc/webapp --readiness --get-url=http://:8080/healthz

# Set an HTTP readiness probe over HTTPS on 127.0.0.1 for a hostNetwork pod
oc set probe dc/router --readiness --get-url=https://127.0.0.1:1936/stats

# Set only the initial-delay-seconds field on all deployments
oc set probe dc --all --readiness --initial-delay-seconds=30
```

6.1.107. oc set resources

使用 pod 模板更新对象上的资源请求/限制

用法示例

```
# Set a deployments nginx container CPU limits to "200m and memory to 512Mi"
oc set resources deployment nginx -c=nginx --limits=cpu=200m,memory=512Mi

# Set the resource request and limits for all containers in nginx
oc set resources deployment nginx --limits=cpu=200m,memory=512Mi --
requests=cpu=100m,memory=256Mi

# Remove the resource requests for resources on containers in nginx
oc set resources deployment nginx --limits=cpu=0,memory=0 --requests=cpu=0,memory=0
```

```
# Print the result (in YAML format) of updating nginx container limits locally, without hitting the server
oc set resources -f path/to/file.yaml --limits=cpu=200m,memory=512Mi --local -o yaml
```

6.1.108. oc set route-backends

更新路由的后端

用法示例

```
# Print the backends on the route 'web'
oc set route-backends web

# Set two backend services on route 'web' with 2/3rds of traffic going to 'a'
oc set route-backends web a=2 b=1

# Increase the traffic percentage going to b by 10%% relative to a
oc set route-backends web --adjust b=+10%%

# Set traffic percentage going to b to 10%% of the traffic going to a
oc set route-backends web --adjust b=10%%

# Set weight of b to 10
oc set route-backends web --adjust b=10

# Set the weight to all backends to zero
oc set route-backends web --zero
```

6.1.109. oc set selector

在资源上设置选择器

用法示例

```
# Set the labels and selector before creating a deployment/service pair.
oc create service clusterip my-svc --clusterip="None" -o yaml --dry-run | oc set selector --local -f - 'environment=qa' -o yaml | oc create -f -
oc create deployment my-dep -o yaml --dry-run | oc label --local -f - environment=qa -o yaml | oc create -f -
```

6.1.110. oc set serviceaccount

更新资源的服务帐户

用法示例

```
# Set deployment nginx-deployment's service account to serviceaccount1
oc set serviceaccount deployment nginx-deployment serviceaccount1

# Print the result (in YAML format) of updated nginx deployment with service account from a local file, without hitting the API server
oc set sa -f nginx-deployment.yaml serviceaccount1 --local --dry-run -o yaml
```

6.1.111. oc set subject

更新角色绑定或集群角色绑定中的用户、组或服务帐户

用法示例

```
# Update a cluster role binding for serviceaccount1
oc set subject clusterrolebinding admin --serviceaccount=namespace:serviceaccount1

# Update a role binding for user1, user2, and group1
oc set subject rolebinding admin --user=user1 --user=user2 --group=group1

# Print the result (in YAML format) of updating role binding subjects locally, without hitting the server
oc create rolebinding admin --role=admin --user=admin -o yaml --dry-run | oc set subject --local -f -
--user=foo -o yaml
```

6.1.112. oc set volumes

更新 pod 模板中的卷

用法示例

```
# List volumes defined on all deployment configs in the current project
oc set volume dc --all

# Add a new empty dir volume to deployment config (dc) 'myapp' mounted under
# /var/lib/myapp
oc set volume dc/myapp --add --mount-path=/var/lib/myapp

# Use an existing persistent volume claim (PVC) to overwrite an existing volume 'v1'
oc set volume dc/myapp --add --name=v1 -t pvc --claim-name=pvc1 --overwrite

# Remove volume 'v1' from deployment config 'myapp'
oc set volume dc/myapp --remove --name=v1

# Create a new persistent volume claim that overwrites an existing volume 'v1'
oc set volume dc/myapp --add --name=v1 -t pvc --claim-size=1G --overwrite

# Change the mount point for volume 'v1' to /data
oc set volume dc/myapp --add --name=v1 -m /data --overwrite

# Modify the deployment config by removing volume mount "v1" from container "c1"
# (and by removing the volume "v1" if no other containers have volume mounts that reference it)
oc set volume dc/myapp --remove --name=v1 --containers=c1

# Add new volume based on a more complex volume source (AWS EBS, GCE PD,
# Ceph, Gluster, NFS, ISCSI, ...)
oc set volume dc/myapp --add -m /data --source=<json-string>
```

6.1.113. oc tag

将现有镜像标记到镜像流中

用法示例

```
  # Tag the current image for the image stream 'openshift/ruby' and tag '2.0' into the image stream
'yourproject/ruby with tag 'tip'
  oc tag openshift/ruby:2.0 yourproject/ruby:tip

  # Tag a specific image
  oc tag
openshift/ruby@sha256:6b646fa6bf5e5e4c7fa41056c27910e679c03ebe7f93e361e6515a9da7e258cc
yourproject/ruby:tip

  # Tag an external container image
  oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip

  # Tag an external container image and request pullthrough for it
  oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip --reference-
policy=local

  # Tag an external container image and include the full manifest list
  oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip --import-
mode=PreserveOriginal

  # Remove the specified spec tag from an image stream
  oc tag openshift/origin-control-plane:latest -d
```

6.1.114. oc version

输出客户端和服务器版本信息

用法示例

```
  # Print the OpenShift client, kube-apiserver, and openshift-apiserver version information for the
current context
  oc version

  # Print the OpenShift client, kube-apiserver, and openshift-apiserver version numbers for the current
context
  oc version --short

  # Print the OpenShift client version information for the current context
  oc version --client
```

6.1.115. oc wait

实验性：等待一个或多个资源上的特定条件

用法示例

```
  # Wait for the pod "busybox1" to contain the status condition of type "Ready"
  oc wait --for=condition=Ready pod/busybox1

  # The default value of status condition is true; you can wait for other targets after an equal delimiter
(compared after Unicode simple case folding, which is a more general form of case-insensitivity):
  oc wait --for=condition=Ready=false pod/busybox1

  # Wait for the pod "busybox1" to contain the status phase to be "Running".
```

```
  oc wait --for=jsonpath='{.status.phase}'=Running pod/busybox1

  # Wait for the pod "busybox1" to be deleted, with a timeout of 60s, after having issued the "delete"
command
  oc delete pod/busybox1
  oc wait --for=delete pod/busybox1 --timeout=60s
```

6.2. OPENSHIFT CLI（OC）管理员命令

6.2.1. oc adm inspect

为给定资源收集调试数据

用法示例

```
  # Collect debugging data for a kubernetes service
  oc adm inspect service/kubernetes

  # Collect debugging data for a node
  oc adm inspect node/<node_name>

  # Collect debugging data for logicalvolumes in a CRD
  oc adm inspect crd/logicalvolumes.topolvm.io

  # Collect debugging data for routes.route.openshift.io in a CRD
  oc adm inspect crd/routes.route.openshift.io
```

6.2.2. oc adm release extract

将更新有效负载的内容提取到磁盘

用法示例

```
  # Use git to check out the source code for the current cluster release to DIR
  oc adm release extract --git=DIR

  # Extract cloud credential requests for AWS
  oc adm release extract --credentials-requests --cloud=aws

  # Use git to check out the source code for the current cluster release to DIR from linux/s390x image
  # Note: Wildcard filter is not supported; pass a single os/arch to extract
  oc adm release extract --git=DIR quay.io/openshift-release-dev/ocp-release:4.11.2 --filter-by-
os=linux/s390x
```

6.2.3. oc adm release info

显示发行版本的信息

用法示例

```
  # Show information about the cluster's current release
  oc adm release info
```

```
# Show the source code that comprises a release
oc adm release info 4.11.2 --commit-urls

# Show the source code difference between two releases
oc adm release info 4.11.0 4.11.2 --commits

# Show where the images referenced by the release are located
oc adm release info quay.io/openshift-release-dev/ocp-release:4.11.2 --pullspecs

# Show information about linux/s390x image
# Note: Wildcard filter is not supported; pass a single os/arch to extract
oc adm release info quay.io/openshift-release-dev/ocp-release:4.11.2 --filter-by-os=linux/s390x
```

6.2.4. oc adm taint

更新节点上的污点

用法示例

```
# Update node 'foo' with a taint with key 'dedicated' and value 'special-user' and effect 'NoSchedule'
# If a taint with that key and effect already exists, its value is replaced as specified
oc adm taint nodes foo dedicated=special-user:NoSchedule

# Remove from node 'foo' the taint with key 'dedicated' and effect 'NoSchedule' if one exists
oc adm taint nodes foo dedicated:NoSchedule-

# Remove from node 'foo' all the taints with key 'dedicated'
oc adm taint nodes foo dedicated-

# Add a taint with key 'dedicated' on nodes having label mylabel=X
oc adm taint node -l myLabel=X  dedicated=foo:PreferNoSchedule

# Add to node 'foo' a taint with key 'bar' and no value
oc adm taint nodes foo bar:NoSchedule
```