Config / Java Environment

JDK

JDK version

JDK 1.0	(January 23, 1996)
JDK 1.1	(February 19, 1997)
J2SE 1.2	(December 8, 1998) - rebranded as Java 2 Standard Edition
J2SE 1.3	(May 8, 2000)
J2SE 1.4	(February 6, 2002)
J2SE 5.0	(September 30, 2004) - originally called J2SE 1.5 but rebranded as 5.0
Java SE 6	(December 11, 2006)
Java SE 7	(July 28, 2011)
Java SE 8	(March 18, 2014)
Java SE 9	(September 21, 2017)
Java SE 10	(March 20, 2018)
Java SE 11	(September 25, 2018) - Long Term Support (LTS)
Java SE 12	(March 19, 2019)
Java SE 13	(September 17, 2019)
Java SE 14	(March 17, 2020)
Java SE 15	(September 15, 2020)
Java SE 16	(March 16, 2021)
Java SE 17	(September 14, 2021) - Long Term Support (LTS)
Java SE 18	(March 22, 2022)
Java SE 19	(September 20, 2022)
Java SE 20	(March 21, 2023)

Command

javac

javac [options] [source files] Compiler, convert the source java file to bytecode class file.

Example:

```
javac HelloWorld.java    =>    HelloWorld.class
```

Options

-d <directory>	Specifies the destination directory for the compiled class files.
-classpath <path> or -cp <path>	Specifies the class path for finding referenced classes.
-sourcepath <path>	Specifies the source code path for finding additional source files.
-encoding <encoding>	Specifies the character encoding used by source files.
-g	Generates all debugging information (local variables, source file, and line numbers).
-g:none	Does not generate any debugging information.
-Xlint	Enables recommended warnings.
-Xlint:<warning>	Enables specific warnings, e.g., -Xlint:unchecked.
-Xdiags:<diagnostic>	Configures diagnostic information, e.g., -Xdiags:verbose for detailed messages.
-deprecation	Shows a description of each use or override of a deprecated member or class.
-source <release>	Provides source compatibility with the specified release.
-target <release>	Generates class files suitable for the specified release.
-verbose	Outputs messages about what the compiler is doing.
-J<flag>	Passes flag directly to the runtime system.

java

Standard Options (Java SE 8)

-version	Displays the version of the JVM.
-help	Displays help information.
-showversion	Displays version information and continues execution.
-server	Selects the server JVM (default on server-class machines).
-client	Selects the client JVM (default on client-class machines).

Memory Management

-Xms=<size>

Sets the initial heap size. Default: 2097152 bytes (2 MB).

-Xmx<size>

Sets the maximum heap size. Default varies by platform and JVM ergonomics.

-XX:NewSize=<size>

Sets the initial size of the young generation (eden). Default: varies.

-XX:MaxNewSize=<size>

Sets the maximum size of the young generation. Default: unlimited.

-XX:+UseAdaptiveSizePolicy

Enable the adaptive size policy for the heap.

-Xss<size>

Sets the thread stack size.

Default varies by platform.

-XX:MaxMetaspaceSize=<size>

Sets the maximum metaspace size.

(introduced in JDK 8, replacing PermSize and MaxPermSize).

-XX:MaxDirectMemorySize=<size>

Set the maximum amount of direct memory that can be allocated by the JVM.

Direct memory is used by NIO buffers and other mechanisms that require off-heap memory allocation.

If this limit is reached, the JVM will throw an OutOfMemoryError when trying to allocate additional direct memory.

-XX:InitialHeapSize=<size>

Sets the initial heap size.

Default: 1/64th of physical memory.

-XX:MaxHeapSize=<size>

Sets the maximum heap size.

Default: 1/4th of physical memory.

-XX:PermSize=<size>

Sets the initial size of the permanent generation.

(Deprecated in JDK 8, replaced by `MaxMetaspaceSize`)

-XX:MaxPermSize=<size>

Sets the maximum size of the permanent generation.

(Deprecated in JDK 8, replaced by `MaxMetaspaceSize`).

Debugging and Monitoring

-XX:HeapDumpPath=<path>
Sets the path to the heap dump file.
-XX:+HeapDumpBeforeFullGC
Generate a heap dump before a Full GC event occurs. This can be useful for analyzing memory usage and identifying potential memory issues.
The dump file can be analyzed using the tool Jvisualvm.

-XX:+HeapDumpOnOutOfMemoryError
Generate a heap dump when an OutOfMemoryError is thrown.
The OOM error can be analyzed using the command jstat.
-XX:+ExitOnOutOfMemoryError
When this option is enabled, if the JVM encounters an `OutOfMemoryError` due to insufficient memory,
it will stop the application immediately instead of continuing to run in an unstable state.
This can help to avoid potential data corruption or further issues that might arise from running the application in a low-memory state.

-Xdebug
Enable debugging.
-Xrunjdwp:<options>
Load the Java Debug Wire Protocol (JDWP) agent for remote debugging.
-verbose:class
Enable verbose output for class loading.
-verbose:gc
Enable verbose output for garbage collection.
-verbose:jni
Enable verbose output for JNI.
-Xprof
Enable CPU profiling.
-javaagent:<jarpath>[=<options>]
specify a Java agent, which is a special type of Java program designed to be loaded at the start of the JVM.
Java agents can instrument Java bytecode at runtime, enabling capabilities such as profiling, monitoring, or even modifying the behavior of Java applications.
Example:
```
java -javaagent:myagent.jar=option1,option2 -jar myapp.jar
```

-XX:+PrintGC
Enables basic GC logging.
-XX:+PrintGCDetails
Print detailed information about each garbage collection event.
-XX:+PrintGCTimeStamps
Adds a date stamp to each line in the GC log.
-XX:+PrintGCApplicationStoppedTime
Print the time that application threads are stopped during garbage collection. (JDK 9 and higher)

-XX:+UseGCLogFileRotation
Enables log file rotation for the GC log.

-XX:NumberOfGCLogFiles=<number>
    Specifies the number of GC log files to keep when using log rotation.
-XX:GCLogFileSize=<size>
    Sets the size limit for each GC log file.
-Xloggc:gc.log
    Directs the JVM to log garbage collection (GC) activity to a specified file.

Performance Tuning

-XX:+AggressiveOpts
    Enable aggressive performance optimization.
-XX:+OptimizeStringConcat
    Enable optimization of string concatenation operations. Default: enabled.
-XX:+UseCompressedOops
    Use compressed references (object pointers). Default: enabled.

Garbage Collection

-XX:+UseSerialGC
    Use the serial garbage collector.
-XX:+UseParallelGC
    Use the parallel garbage collector.
-XX:+UseConcMarkSweepGC
    Use the concurrent mark-sweep garbage collector.
-XX:+UseG1GC
    Use the Garbage-First garbage collector.
    (became the default in JDK 9).
-XX:NewRatio=<ratio>
    Sets the ratio between young and old generation sizes.
-XX:SurvivorRatio=<ratio>
    Sets the ratio between eden and survivor spaces in the young generation.
-XX:MaxTenuringThreshold=<threshold>
    Sets the maximum tenuring threshold for the young generation.

-XX:+UnlockExperimentalVMOptions
    (JDK 9 and higher)
-XX:+UseZGC
    (Introduced as experimental in JDK 11, refined in JDK 15+.
-XX:+UseShenandoahGC
    (Introduced in JDK 12, refined in later versions)
-XX:+UseEpsilonGC
    (Introduced in JDK 11, available in later versions)

Class Loading and Execution

-classpath <path> or -cp <path>
    Specifies the class search path.
-D<name>=<value>
    Sets a system property.
-jar <file>
    Executes a program encapsulated in a JAR file.

Exmaple:

```
nohup java [options] -jar /home/match.jar >> nohup.out 2>&1 &
```

-ea or -enableassertions

Enable assertions.

-da or -disableassertions

Disable assertions.

-esa or -enablesystemassertions

Enable system assertions.

-dsa or -disablesystemassertions

Disable system assertions.

Security

-Djava.security.manager

Enables the security manager.

-Djava.security.policy=<url>

Specifies the security policy file.

--add-opens <source-module>/<package>=<target-module>

Allow deep reflection access to specific internal packages that would otherwise be inaccessible due to strong encapsulation in the module system.

This is part of Java's module system (JPMS - Java Platform Module System), introduced in Java 9.

// java --add-opens java.base/java.nio=ALL-UNNAMED -jar myapp.jar

Opens the java.nio package in the java.base module (which contains core Java classes like ByteBuffer, FileChannel, etc.).

Allows unnamed modules (code on the classpath, like most libraries) to use reflection on java.nio internals.

Databricks ERROR:

Without this parameter, the class com.databricks.client.spark.arrow.ArrowBuffer may not be accessible.

Related class: org.apache.arrow.memory.RootAllocator

JIT Compiler

-Xcomp Compile methods on first invocation.

-Xint Interpret all bytecode (disable JIT compilation).

-Xmixed Mixed mode execution (default).

-XX:ReservedCodeCacheSize=<size>

Sets the code cache size.

-XX:+UseCodeCacheFlushing

Enables code cache flushing.

-XX:+TieredCompilation

Enables tiered compilation.

Miscellaneous

-Xbatch

Disable background compilation.

-Xnoclassgc

Disable class garbage collection.

-Xloggc:<file>

Log GC status to a file.

Application Startup

-Djava.awt.headless=true

Enables headless mode in the AWT subsystem.

-Duser.timezone=GMT+8

Sets the default time zone for the JVM.java

-Djava.rmi.server.hostname=192.168.16.237

Specifies the hostname/IP address that the RMI server will bind to and advertise to clients.

-Dcom.sun.management.jmxremote=true

Enables JMX remote management and monitoring capabilities.

-Dcom.sun.management.jmxremote.port=1099

Specifies the port number for the JMX remote management agent to listen on.

-Dcom.sun.management.jmxremote.rmi.port=1099

Specifies the RMI port for the JMX remote management agent.

-Dcom.sun.management.jmxremote.authenticate=false

Disables authentication for the JMX remote management agent.

-Dcom.sun.management.jmxremote.ssl=false

Disables SSL for the JMX remote management agent.

jps

The jps command is a utility provided by the Java Development Kit (JDK) that lists Java processes running on a machine.
It stands for "Java Virtual Machine Process Status Tool" and provides information such as the process ID (PID) and the main class or JAR file being executed by each Java process.

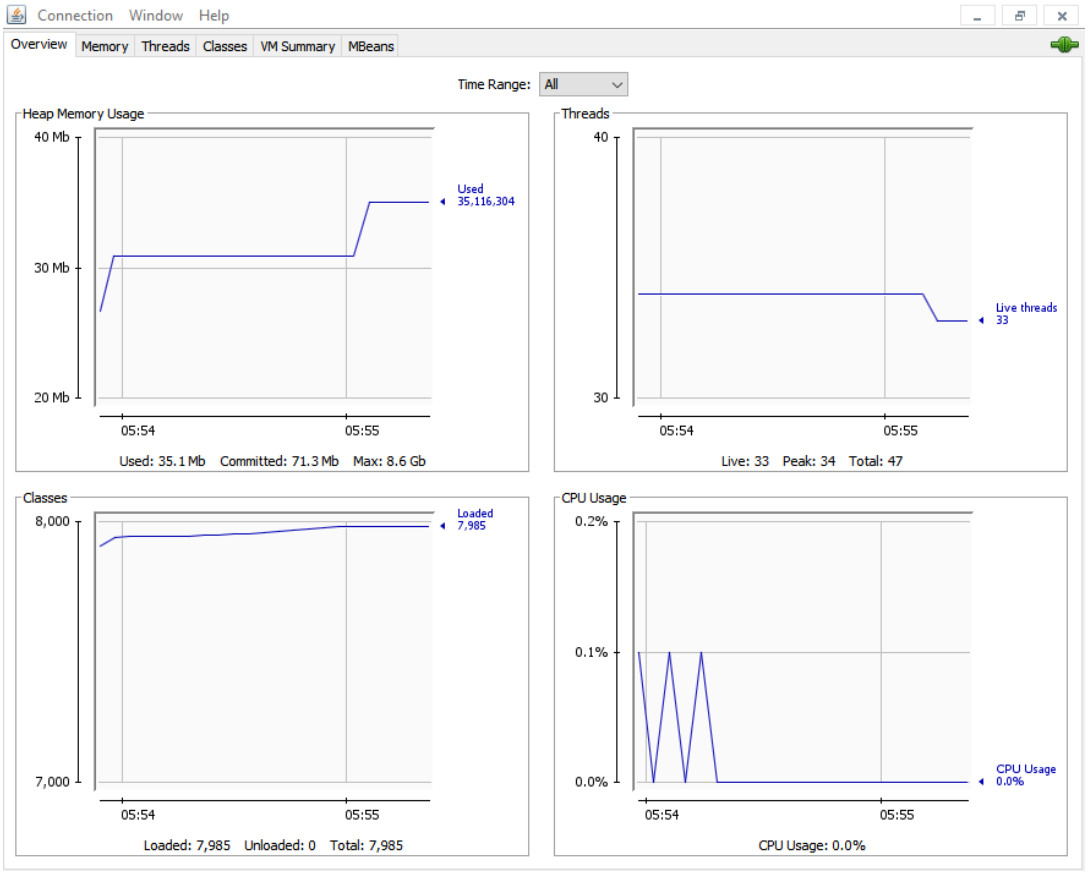
jps [options]	The basic syntax for using the jps command
jps	To list all Java processes running on the local machine

Options

-l	Displays the full package name for the main class or JAR file being executed.
-m	Displays the arguments passed to the main method.
-v	Displays the arguments passed to the JVM.
-q	Displays only the PID (process ID) of the Java processes.
-J<flag>	Passes flag directly to the JVM running jps.
-V	Displays version information.
-h	Displays help information.

jconsole

The jconsole command launches Java Monitoring and Management Console,
a graphical user interface tool that provides information about the performance and resource consumption of applications running on the Java Virtual Machine (JVM).
It utilizes Java Management Extensions (JMX) to monitor and manage Java applications.



jconsole	To monitor a local Java application, simply run jconsole without any arguments:
jconsole <hostname>:<port>	To monitor a remote Java application, specify the JMX URL of the remote application:

Example:

```
jconsole 192.168.1.100:9999
```

Options

-interval=<seconds>	Sets the update interval for the monitoring data in seconds. Default is 4 seconds.
-pluginpath <path>	Specifies a directory or JAR file from which JConsole loads user-defined plugins.
-J<flag>	Passes the specified flag to the Java runtime environment.
-version	Displays the version information of the JConsole utility.
-help	Displays help information.

jmap

The jmap command is a utility provided by the Java Development Kit (JDK) for generating heap dumps and obtaining information about the Java heap and memory usage of a running Java process.

It stands for "Java Memory Map" and is commonly used for diagnosing memory-related issues such as memory leaks and excessive memory consumption.

jmap [option] <pid>	The basic syntax for using the jmap command

Example:

```
jmap -dump:file=heapdump.bin 12345
```

Options

-dump:live,format=b	Generates a heap dump in binary format (b) containing only live objects.
-dump:file=<file>	Specifies the file name for the heap dump.
-heap <pid>	Prints heap summary information.
-histo[:live] <pid>	Prints a histogram of the heap, optionally including only live objects.
-F	Forces the heap dump, even if the target process is unresponsive.
-h	Displays help information.

jstack

The jstack command is a utility provided by the Java Development Kit (JDK) for generating Java thread stack traces of a running Java process.

It is used for diagnosing thread-related issues such as deadlocks, high CPU usage, and thread contention.

By examining the thread dump, you can determine the current execution location of each thread and understand what the application is doing at a particular point in time.

jstack [option] <pid> The basic syntax for using the jstack command.

Example:

```
jstack 12345
```

Options

-l Prints more information about locks and monitors, including owned monitors.

-F Forces a stack dump of the target process, even if it is unresponsive.

-m Prints mixed mode (native and Java) stack traces.

-h Displays help information.

Output

"Async-Craig-3" #51 prio=5 os_prio=0 cpu=6765.62ms elapsed=6.90s tid=0x000001a8c925e570 nid=0x4a74 runnable [0x00000060f3dfd000]

java.lang.Thread.State: RUNNABLE

at com.simi.service.LongRunningService.performLongRunningTask(LongRunningService.java:71)

at jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(java.base@17.0.8/Native Method)

at jdk.internal.reflect.NativeMethodAccessorImpl.invoke(java.base@17.0.8/NativeMethodAccessorImpl.java:77)

at jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(java.base@17.0.8/DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(java.base@17.0.8/Method.java:568)

at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:343)

at org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:196)

at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:163)

at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750)

at org.springframework.aop.interceptor.AsyncExecutionInterceptor.lambda$invoke$0(AsyncExecutionInterceptor.java:115)

at org.springframework.aop.interceptor.AsyncExecutionInterceptor$$Lambda$796/0x000001a88448a458.call(Unknown Source)

at org.springframework.util.concurrent.FutureUtils.lambda$toSupplier$0(FutureUtils.java:74)

at org.springframework.util.concurrent.FutureUtils$$Lambda$797/0x000001a88448a878.get(Unknown Source)

at java.util.concurrent.CompletableFuture$AsyncSupply.run(java.base@17.0.8/CompletableFuture.java:1768)

at java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@17.0.8/ThreadPoolExecutor.java:1136)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@17.0.8/ThreadPoolExecutor.java:635)

at java.lang.Thread.run(java.base@17.0.8/Thread.java:833)

"main" #1 prio=5 os_prio=0 tid=0x00007fcd98001800 nid=0x1c03 runnable [0x00007fcd98bfc000]

- The thread name is " Async-Craig-3".
- #51 indicates it is the 51st thread.
- prio=5 indicates the priority of the thread.
- runnable indicates the state of the thread.

- tid and nid are the thread ID and native thread ID, respectively.
- The thread is currently executing and the call stack is shown below.

jstat

The jstat command is a Java Virtual Machine (JVM) statistics monitoring tool. It provides information about the performance and resource consumption of an active JVM. Here's a guide on how to use jstat.

jstat [option] <vmid> [interval] [count]

option: Specifies the type of statistics to monitor.
vmid: The virtual machine identifier. This is the process ID of the JVM.
interval: The time between each sample (in milliseconds).
count: The number of samples to display.

Common Options

Here are some of the most commonly used options:

-class	Displays class loader statistics.
-gc	Displays garbage collection statistics.
-gccapacity	Displays the capacities of the generations and their corresponding spaces.
-gcutil	Displays a summary of garbage collection statistics.
-gcnew	Displays statistics about the behavior of the new generation.
-gcnewcapacity	Displays statistics about the sizes of the new generation and its spaces.
-gcold	Displays statistics about the behavior of the old and permanent generations.
-gcoldcapacity	Displays statistics about the sizes of the old and permanent generations.
-gcpermcapacity	Displays statistics about the sizes of the permanent generation.
-compiler	Displays statistics of the behavior of the HotSpot Just-in-Time (JIT) compiler.
-printcompilation	Displays Java method compilation statistics.

Interpreting jstat Output

The output of jstat will vary depending on the option used. Here's a brief overview of what some of the columns mean:

S0: Percentage of Survivor space 0 used.
S1: Percentage of Survivor space 1 used.
E: Percentage of Eden space used.
O: Percentage of Old generation space used.
M: Percentage of Metaspace used.
CCS: Compressed Class Space utilization.
YGC: Number of young generation garbage collections.
YGCT: Time (in seconds) spent in young generation garbage collections.
FGC: Number of full garbage collections.
FGCT: Time (in seconds) spent in full garbage collections.
CGC: Number of cycles of concurrent garbage collections (if applicable).
CGCT: Time (in seconds) spent in concurrent garbage collections (if applicable).
GCT: Total time (in seconds) spent in garbage collections (both young and full).

Usage

jstat -gcutil $pid 1000

```
(base) ➜  ~ jstat -gcutil 99242 1000
  S0     S1     E      O      M     CCS    YGC     YGCT    FGC    FGCT     CGC    CGCT     GCT
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
  0.00  32.44   9.59   0.05  95.95  89.82      1    0.003     0    0.000     -         -    0.003
```

S0: 0% of Survivor space 0 is used.
S1: 32.44% of Survivor space 1 is used.
E: 9.59% of Eden space is used.
O: 0.05% of the Old generation is used.
P: 91.45% of the Permanent generation is used.
YGC: There have been 4245 young generation garbage collections.
YGCT: 35.610 seconds have been spent in young generation garbage collections.
FGC: There have been 9 full garbage collections.
FGCT: 2.001 seconds have been spent in full garbage collections.
GCT: 37.611 seconds have been spent in total garbage collections.

keytool

The keytool command is a utility provided by the Java Development Kit (JDK) for managing cryptographic keys, certificates, and keystores.
It is commonly used for generating, importing, exporting, and managing keys and certificates for secure communication in Java applications.

keytool [options] [commands]
The basic syntax for using the keytool command
Example:

```
// To generate a self-signed certificate and store it in a keystore file
keytool -genkeypair -alias mykey -keyalg RSA -keysize 2048 -validity 365 -keystore keystore.jks
```

Options

-genkeypair	Generates a key pair (public/private key).
-alias <alias>	Specifies the alias for the key entry.
-keyalg <alg>	Specifies the key algorithm (e.g., RSA, DSA).
-keysize <size>	Specifies the key size in bits.
-validity <days>	Specifies the validity period of the certificate in days.
-keystore <file>	Specifies the keystore file name.
-storepass <pass>	Specifies the password for the keystore.
-storetype <type>	Specifies the type of keystore (e.g., JKS, PKCS12).
-list	Lists entries in a keystore.
-importcert	Imports a certificate or certificate chain.

Example:

```
keytool -importcert -alias your.alias -keystore cacerts.jks -file certificate.pem
-storepass password
```

-exportcert	Exports a certificate.

-help Displays help information.

openssl

x509

The openssl x509 command is a utility provided by OpenSSL for working with X.509 certificates. It allows you to view, manipulate, and manage X.509 certificates in various formats.

openssl x509 [options] [certfile] The basic syntax for using the openssl x509 command

Options

-in <file> Specifies the input certificate file.
-out <file> Specifies the output file.
-text Prints a text representation of the certificate.
Example:
```
openssl x509 -in xxx.crt -text
```
-noout Prevents output of the certificate.
-subject Prints the subject of the certificate.
-issuer Prints the issuer of the certificate.
-startdate Prints the start date of the certificate validity.
-enddate Prints the end date of the certificate validity.
-modulus Prints the modulus of the public key.
-fingerprint Prints the fingerprint of the certificate.
-serial Prints the serial number of the certificate.
-inform <format> Specifies the input format (PEM, DER, etc.).
-outform <format> Specifies the output format (PEM, DER, etc.).
-help Displays help information.

Example Usage

1. DER to PEM
   - concert a DER encoded certificate into a PEM encoded certificate
     ```
     openssl x509 -inform DER -in xxx.der -outform PEM -out xxx.crt
     openssl x509 -inform DER -in xxx.cer -outform PEM -out xxx.pem
     openssl x509 -in xxx.crt -out xxx.pem
     ```
   - concert a DER encoded private key into a PEM encoded private key
     ```
     openssl rsa -inform DER -in xxx.der -outform PEM -out xxx.key
     ```
2. PEM to DER
   - concert a PEM encoded certificate into a DER encoded certificate
     ```
     openssl x509 -inform PEM -in xxx.crt -outform DER -out xxx.der
     ```
   - concert a PEM encoded private key into a DER encoded private key
     ```
     openssl rsa -inform PEM -in xxx.key -outform DER -out xxx.der
     ```

req

The openssl req command is used to generate and manage X.509 certificate signing requests (CSRs) and related operations. It allows you to create CSRs, sign them, and perform various tasks related to certificate requests and certificate authorities.

openssl req [options] [arguments] The basic syntax for using the openssl req command
Example:
```
openssl req -new -newkey rsa:2048 -keyout private.key -out csr.pem
```

Options

-new Generate a new CSR.
-newkey <algo> Generate a new private key.
-keyout <file> Specify the output file for the private key.

-out <file>	Specify the output file for the CSR.
-key <file>	Specify the input file for the private key.
-subj <arg>	Specify the subject information for the CSR.
-config <file>	Specify the configuration file.
-nodes	Do not encrypt the private key.
-days <n>	Specify the number of days the CSR is valid for.
-batch	Do not prompt for certificate details.
-verify	Verify the CSR against its own signature.
-text	Print a text representation of the CSR.
-noout	Do not output the CSR or private key.
-help	Display help information.

Example Usage

1. Generate CSR with New Private Key

   ```
   openssl req -new -newkey rsa:2048 -keyout private.key -out csr.pem
   ```

   This command generates a new private key and creates a CSR, saving the private key to private.key and the CSR to csr.pem.

2. Generate CSR with Existing Private Key

   ```
   openssl req -new -key existing.key -out csr.pem
   ```

   This command creates a CSR using an existing private key stored in existing.key and saves the CSR to csr.pem.

3. Generate CSR with Subject Information

   ```
   openssl req -new -key private.key -out csr.pem -subj "/C=US/ST=California/L=San Francisco/O=Example Corp/CN=example.com"
   ```

   This command creates a CSR with specified subject information (country, state, locality, organization, common name) and saves it to csr.pem.

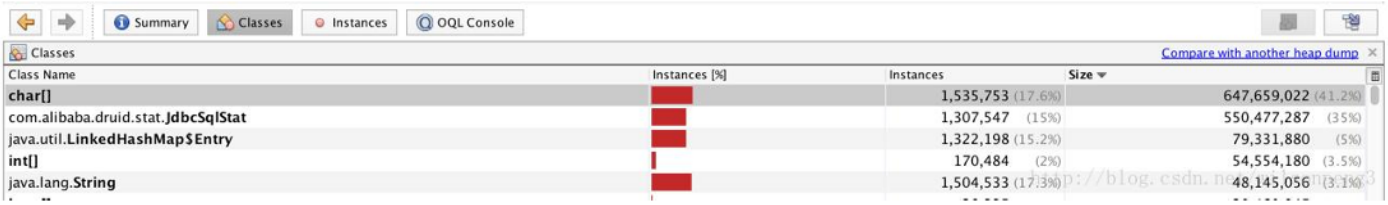
4. Print CSR Details

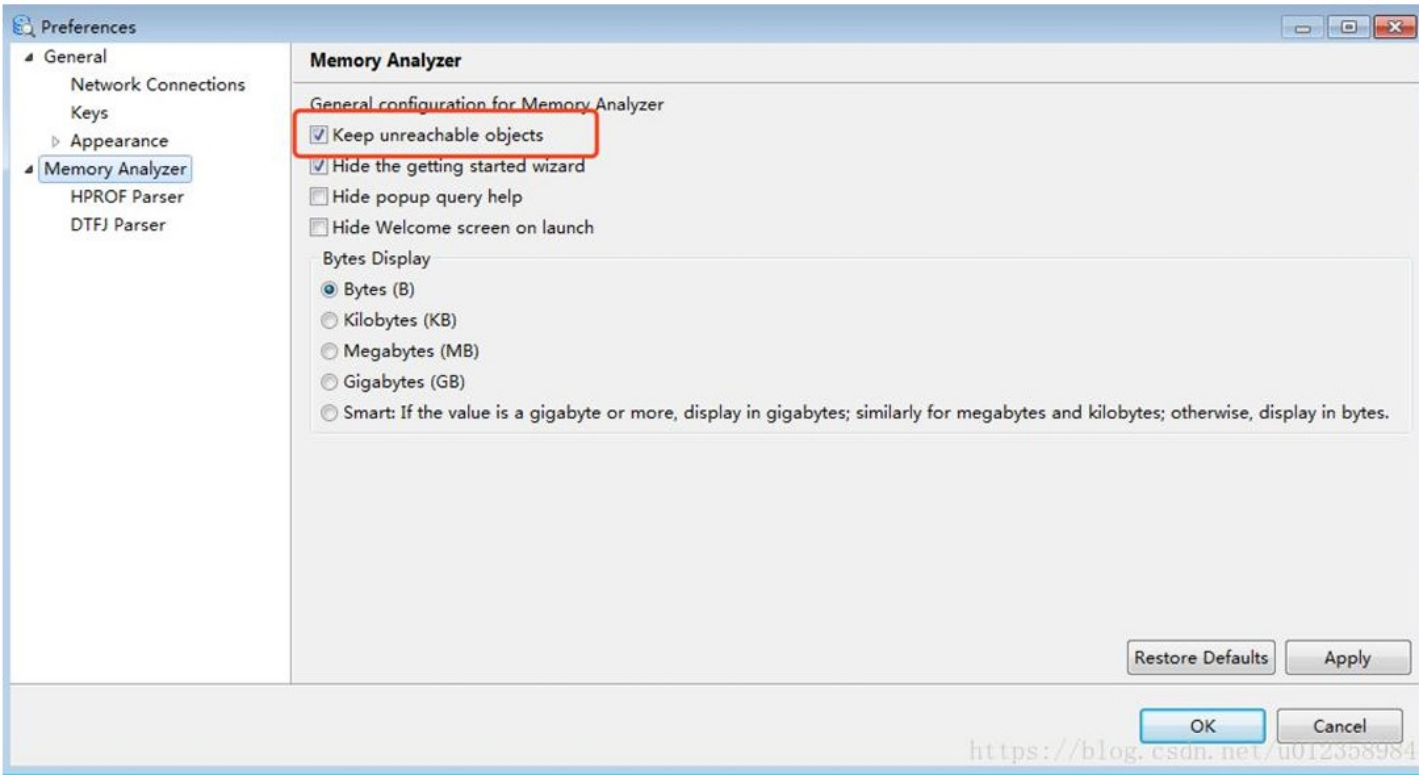
   ```
   openssl req -in csr.pem -text -noout
   ```

   This command prints a text representation of the CSR stored in csr.pem.

Analysis Dump File

jvisualvm





Windows Installation

1. 下载

https://www.oracle.com/java/technologies/downloads/#jdk20-windows

2. msi 引导安装，新建目录 D:\JAVA\jdk1.8.0 D:\JAVA\jre1.8.0

2. 配置环境变量

JDK 环境变量 %JAVA_HOME%\bin %JAVA_HOME%\jre\bin %JAVA_HOME%\lib %JAVA_HOME%\lib\tools.jar

JRE 环境变量 %JRE_HOME% D:\JAVA\jre1.8.0

ERROR

require '1.7' not '1.8'(安装新版本java错误) 新安装JDK的java.exe javaw.exe javaws.exe替换复制到 C:\Windows\system32目录

修改注册表 HEKY_LOCAL_MACHINE -- SOFTWARE -- JavaSoft -- JavaRuntimeEnvironment CurrentVersion=1.8.xxx JavaFamilVersion=1.8.xxx

%JAVA%不生效 System32 优先级高于配置的%JAVA%目录，这两个目录都有 java.exe 等文件，需要直接在 Path 内配置 java 环境变量

Linux Installation

local 安装>>

1. 拉取 jdk8 压缩包： wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz
1. 拉取 jdk11 压缩包：wget http://repo.huaweicloud.com/openjdk/11.0.1/openjdk-11.0.1_linux-x64_bin.tar.gz
2. 解压缩：tar -xzvf jdk-8u131-linux-x64.tar.gz -C /usr/local （ /usr/local 等同/usr/local/ ）
3. 添加 JDK 环境变量： export JAVA_HOME=/usr/local/jdk1.8.0_321 （ssh 有时登录非交互终端，访问不到/etc/profile，需要添加到.bashrc 中）

export PATH=$PATH:$JAVA_HOME/bin

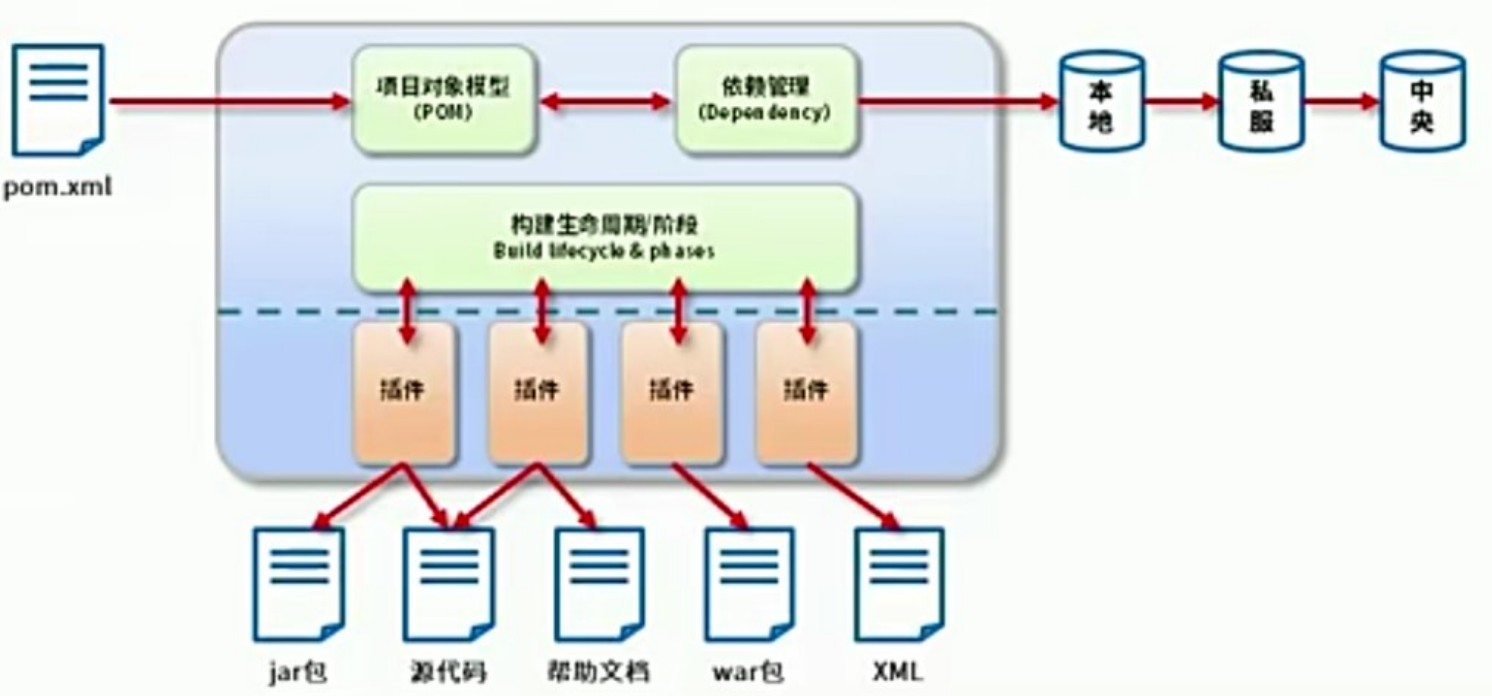
yum 安装>>

yum list java-1.8.0-openjdk*

maven
Core

Maven：Maven 的本质是一个项目管理工具，将项目开发和管理过程抽象成一个项目对象模型（POM）

Maven 中的坐标用于描述仓库中资源的位置



Command Execution

Maven has a predefined set of plugins bound to specific phases of its lifecycle.

For the compile phase, the Maven Compiler Plugin is the default plugin that Maven uses.

If you don't specify the plugin explicitly in your pom.xml, Maven uses its default configuration for the maven-compiler-plugin.

Default Plugin Version

If you don't define the version of the maven-compiler-plugin,

Maven attempts to use the version defined by the Maven Super POM.

The Super POM is the default parent POM for all Maven projects.

It contains default settings, including a default version for many plugins.

Import Dependencies

Use local dependencies

If the version of the current dependency remains unchanged, Maven will use the same JAR file from the local repository.

The dependency will be re-downloaded only after it is deleted in local repository.

Fetch the latest version

- When the version in the pom.xml file is updated
- When the local repository is cleared or deleted

  If the JAR file for a dependency is deleted from the local repository (usually in the .m2 directory), Maven will check the remote repository for the latest version specified in the pom.xml.
- When the maven-dependency-plugin or maven-updates-plugin is used

  These plugins can be used to force Maven to check for newer versions of dependencies by running specific

commands (such as mvn versions:display-dependency-updates).

- When the snapshot version is used

  If a dependency is using a SNAPSHOT version, Maven will always check for the latest snapshot release from the remote repository unless it is explicitly instructed not to.

- When running with the -U (update) flag

  You can use the -U flag when running a Maven build (mvn clean install -U), which forces Maven to check for updated dependencies even if the local version exists.

Versions

Behavior of Snapshot Versions

Latest Available Snapshot

Maven will always check for the latest snapshot version from the remote repository.

For example, if you have a 1.0-SNAPSHOT version in your pom.xml, and a new version of the snapshot (e.g., 1.0-20231116.123456-15) is released,

The 1.0-SNAPSHOT version is essentially a placeholder for the latest build in a development stream, and it's not tied to a specific version number.

Maven will download that newer snapshot, provided that the local repository doesn't already contain it.

```
<dependency>
    <groupId>com.example</groupId>
    <artifactId>my-library</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

Frequent Updates

Snapshot versions are typically updated frequently, as they represent ongoing development. As such, when using snapshot versions,

Maven frequently checks for new builds during each build cycle, unlike stable release versions, which are only updated when the version number is incremented.

Snapshot Handling

Remote Repository Check

If you run a Maven build with a snapshot dependency,

Maven will check the remote repository for any newer snapshot versions, even if it already has a cached snapshot in the local repository.

Timestamped Versions

Snapshot versions often include timestamps in their filenames, like 1.0-20231116.123456-15.

This ensures that Maven can differentiate between different versions of a snapshot and fetch the latest one.

Dependency Mediation

Maven uses a "nearest wins" strategy to resolve conflicts.

When multiple versions of the same dependency are present, the version that is nearest to the top of the dependency tree (the root project) is used.

Managing Dependency Versions

Dependency Management Section

You can use the <dependencyManagement> section in the pom.xml file of a parent project.

This section allows you to specify the version of dependencies that all child modules should use.

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.example</groupId>
            <artifactId>my-library</artifactId>
            <version>1.2.3</version>
```

```
            </dependency>
        </dependencies>
    </dependencyManagement>
```

This does not actually include the dependency in the build but enforces the specified version if a child module declares a dependency on com.example:my-library.

Explicit Version Declaration

You can explicitly declare the version of a dependency in your project's <dependencies> section to ensure the desired version is used.

```
<dependencies>
    <dependency>
        <groupId>com.example</groupId>
        <artifactId>my-library</artifactId>
        <version>1.2.3</version>
    </dependency>
</dependencies>
```

Exclusions

If a transitive dependency brings in an unwanted version of a library, you can exclude it.

```
<dependencies>
    <dependency>
        <groupId>com.example</groupId>
        <artifactId>my-dependency</artifactId>
        <version>1.0.0</version>
        <exclusions>
            <exclusion>
                <groupId>com.example</groupId>
                <artifactId>unwanted-library</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
</dependencies>
```

Using Dependency Scope

Dependency scopes can help manage dependencies by limiting their visibility and usage. For instance, a dependency marked with scope as test will only be used for testing.

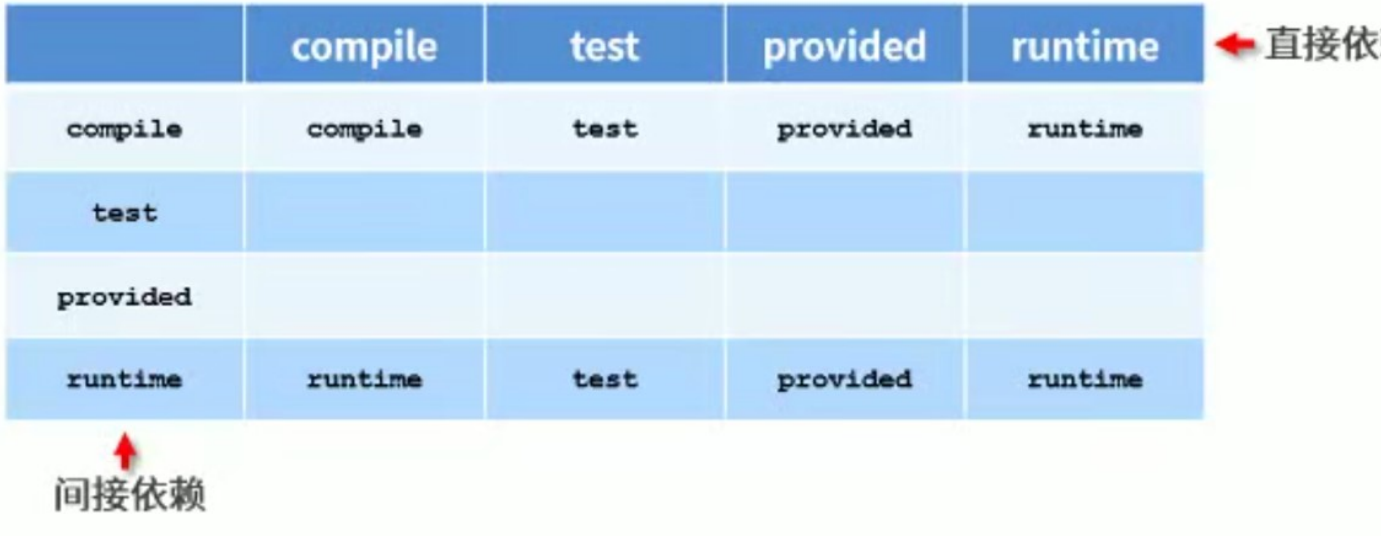
```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Dependencies

依赖合并：

scope	主代码	测试代码	打包	范例
compile(默认)	Y	Y	Y	log4j
test		Y		junit
provided	Y	Y		servlet-api
runtime			Y	jdbc

依赖传递合并范围：



Blackduck：找出 issue 的依赖 => 尝试直接升级 => 尝试间接升级父依赖（有些父依赖会意识到问题，并排除风险项）

Phase (Maven Plugin)

生命周期:生命周期阶段需要绑定到某个插件的目标才能完成真正的工作(test 阶段正是与 maven-surefire-plugin 的 test 目标相绑定了，这是一个内置的绑定)

clean 清理工作

pre-clean 执行一世需要在 clean.之前完成的工作

clean 移除所有上一次构建生成的文件

post-clean 执行一些需要在 clean 之后立刻完成的工作

default 核心工作（编译，测试，打包，部署）

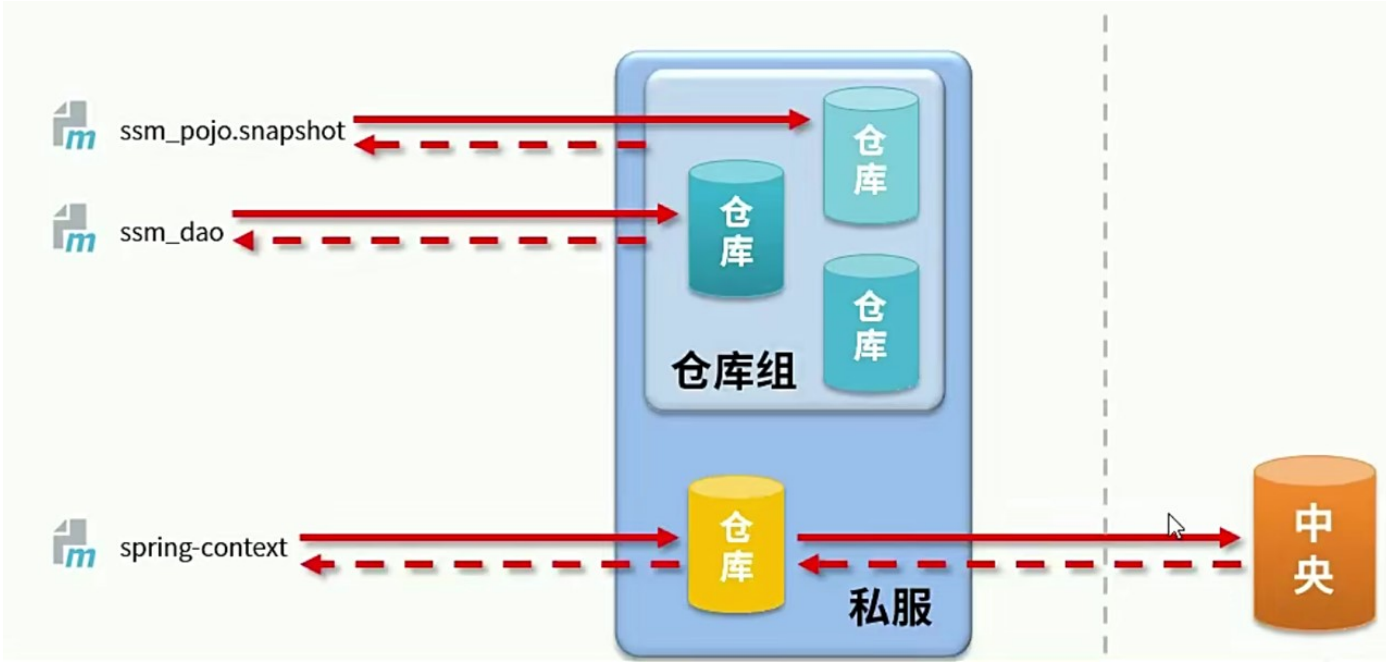
maven 插件与生命周期内的阶段绑定，在执行到对应生命周期时执行对应的插件功能，默认 maven 在各个生命周期上绑定有预设的功能，通过 pow.xml 内定义的插件可以自定义其他功能

validate（校验）	校验项目是否正确并且所有必要的信息可以完成项目的构建过程。
initialize（初始化）	初始化构建状态，比如设置属性值。
generate-sources（生成源代码）	生成包含在编译阶段中的任何源代码。
process-sources（处理源代码）	处理源代码，比如说，过滤任意值。
generate-resources（生成资源文件）	生成将会包含在项目包中的资源文件。
process-resources （处理资源文件）	复制和处理资源到目标目录，为打包阶段最好准备。
compile（编译）	编译项目的源代码。
process-classes（处理类文件）	处理编译生成的文件，比如说对Java class文件做字节码改善优化。
generate-test-sources（生成测试源代码）	生成包含在编译阶段中的任何测试源代码。
process-test-sources（处理测试源代码）	处理测试源代码，比如说，过滤任意值。
generate-test-resources（生成测试资源文件）	为测试创建资源文件。
process-test-resources（处理测试资源文件）	复制和处理测试资源到目标目录。
test-compile（编译测试源码）	编译测试源代码到测试目标目录.
process-test-classes（处理测试类文件）	处理测试源码编译生成的文件。
test（测试）	使用合适的单元测试框架运行测试（Juint是其中之一）。
prepare-package（准备打包）	在实际打包之前，执行任何的必要的操作为打包做准备。
package（打包）	将编译后的代码打包成可分发格式的文件，比如JAR、WAR或者EAR文件。
pre-integration-test（集成测试前）	在执行集成测试前进行必要的动作。比如说，搭建需要的环境。
integration-test（集成测试）	处理和部署项目到可以运行集成测试环境中。
post-integration-test（集成测试后）	在执行集成测试完成后进行必要的动作。比如说，清理集成测试环境。
verify （验证）	运行任意的检查来验证项目包有效且达到质量标准。
install（安装）	安装项目包到本地仓库，这样项目包可以用作其他本地项目的依赖。
deploy（部署）	将最终的项目包复制到远程仓库中与其他开发者和项目共享。

site　　产生报告，发布站点

pre-site	执行一些需要在生成站点文档之前完成的工作
site	生成项目的站点文档
post-site	执行一些需要在生成站点文档之后完成的工作，并且为部署做准备
site-deploy	将生成的站点文档部署到特定的服务器上

Repository



宿主仓库 hosted（常用）

保存无法从中央仓库获取的资源（自定义）

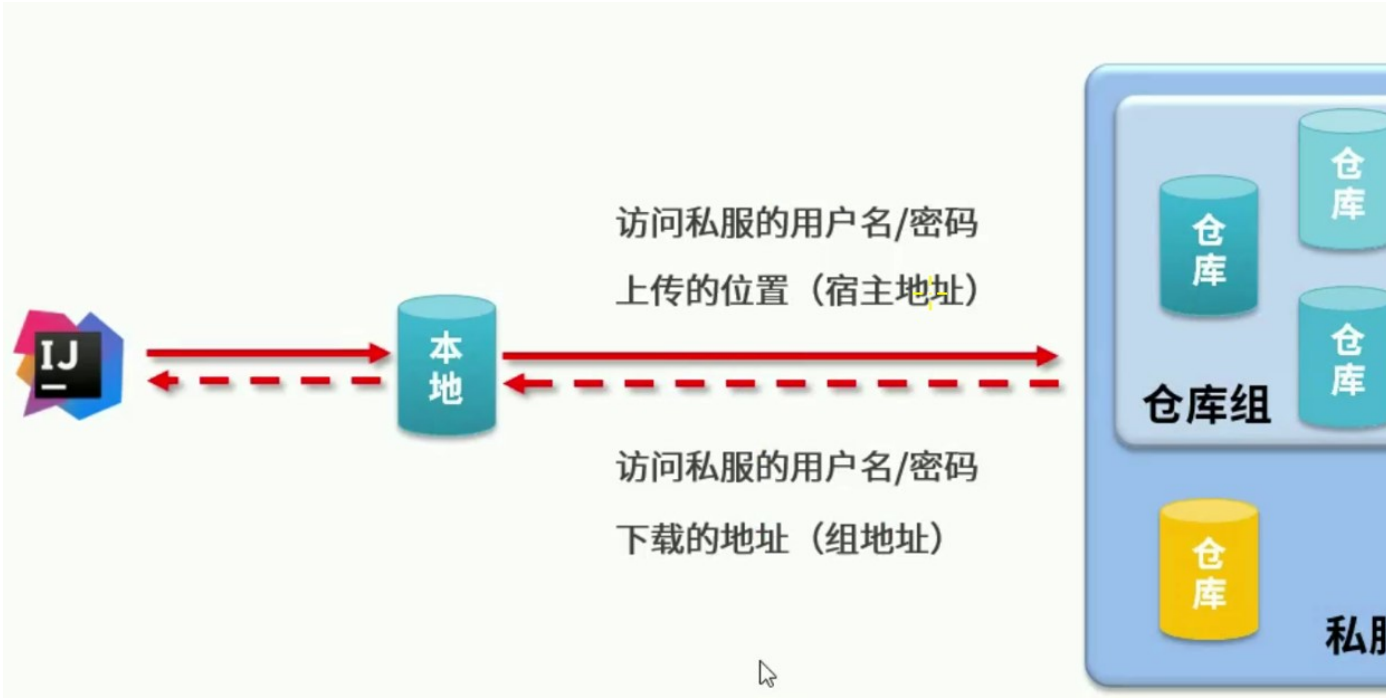
代理仓库 proxy

代理远程仓库，通过 nexus 访问其他公共仓库，例如中央仓库

仓库组 group

将若干个仓库组成一个群组，简化配置

仓库组不能保存资源，属于设计型仓库



pow.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

Private Configuration

<parent>
When using the Spring Boot parent project, if the dependencies in a child project fail to import, you can first install the root project and then install the child project.
This ensures that the child project uses the dependency versions defined in the parent project, avoiding version conflicts.
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.4.4</version>
<relativePath/>
Helps Maven locate the parent module's POM file.
If this configuration is omitted, Maven will default to searching for the parent project's pom.xml in the parent directory.
</parent>
<modelVersion>4.0.0</modelVersion>
Pom object version
<name>multisdk</name>
<artifactId>multisdk</artifactId>
<packaging>pom</packaging>
The packaging type of the project：pom、jar、war
Default Value: jar
The pom project, as a parent project, does not have Java code and does not execute any code, only for the purpose of aggregating projects or passing dependencies

<modules>
<module>common</module>
<module>function-one</module>
<module>function-two</module>
</modules>

<profiles> 区分多环境下的不同配置
<profile> 定义具体的环境：生产环境
<id>pro_env</id> 定义环境对应的唯一名称
<properties> 定义环境中换用的属性值
<jdbc.url>jdbc:mysql://127.1.1.1:3306 /ssm_db</jdbc.url>
</properties>
<activation> 设置默认启动
<activeByDefault>true</ activeByDefault>
</activation>
</profile>
<profile> 定义具体的环境：开发环境
<id>dep_env</id>
<properties>
<jdbc.url>jdbc:mysql://127.2.2.2:3306/ssm_db</jdbc.url>
</properties>
</profile>
</profiles>

Inheriable Configuration

<groupId>com.sdk</groupId>　　maven 项目所属组织名称

<version>1.0-SNAPSHOT</version>　　项目版本

SNAPSHOT（快照版本）　　构建模块输出的临时性版本，也是测试阶段版本

RELEASE（发布版本）　　向团队外部发布较为稳定的版本。

5.1.9.RELEASE　　<主版本>.<次版本>.<增量版本>.<里程碑版本>

主版本：表示项目重大架构的变更，如: spring5 相较于 spring4 的迭代

次版本：表示有较大的功能增加和变化，或者全面系统地修复漏洞

增量版本：表示有重大漏洞的修复

里程碑版本：表明一个版本的里程碑（版本内部）。这样的版本同下一个正式版本相比，相对来说不是很稳定，有待更多的测试

<description>xxx</description>　　项目描述信息

<organization>xxx</organization>　　项目的组织信息

<inceptionYear >　　项目的创始年份

<url>　　项目的 URL 地址

<developers　　项目的开发者信息

<contributors>　　项目的贡献者信息

<issueManagement>　　项目的缺陷跟踪系统信息

<ciManagement>　　项目的持续集成系统信息

<scm>　　项目的版本控制系统信息

<malilingLists>　　项目的邮件列表信息

properties

<properties>

Common Properties:

Built-in Properties:

${basedir}　　The root directory of the current project.

${version}　　The current project's version.

${project.basedir}　　The project's root directory.

${project.build.directory}　　The build directory, default is `target`.

${project.build.outputDirectory}　　The build output directory, default is `target/classes`.

${project.build.finalName}　　The packaged file name, default format is `{project.artifactId}-{project.version}`.

${project.packaging}　　The packaging type, default is `jar`.

${project.xxx}　　Content from any node of the current `POM` file.

${project.parent.version}　　The version of the parent project.

Setting Properties:

${settings.localRepository}　　The local repository path defined in Maven's `settings.xml`.

${env.JAVA_HOME}　　The `JAVA_HOME` environment variable.

(Supported environment variables can be viewed by running `mvn help:system`.)

JAVA System Properties:

${user.home}　　The current user's home directory.

(Supported system properties can be viewed by running `mvn help:system`.)

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

```
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <spring-cloud.version>Hoxton.SR11</spring-cloud.version>
    <java.version>1.8</java.version>

    <maven.compiler.source>${java.version}</maven.compiler.source>
        Set the version in compiler plugin properties or directly configure in maven-compile-plugin
    <maven.compiler.target> ${java.version} </maven.compiler.target>

    <spring-boot.version>2.0.3.RELEASE</spring-boot.version>
    <spring.framework.security.version>5.2.4.RELEASE</spring.framework.security.version>

    <destDir>${project.build.outputDirectory}/META-INF/resources/</destDir>
</properties>
```

dependencies

```
<dependencies>
        <dependency>                聚合的子工程
            <groupId>com.jd</groupId>
            <artifactId>demo-base</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-acl</artifactId>
        <version>${spring.framework.security.version}</version>      导入 properties 属性内定义的版本
    </dependency>
</dependencies>
```

dependencyManagement

<dependencyManagement>　　子工程默认依赖版本和配置（父工程使用）（子项目可省略版本号，Maven 会沿着父子层次向上走寻找对应版本，直到找到一个拥有 dependencyManagement 元素的项目，然后它就会使用在这个 dependencyManagement 元素中指定的版本号）

```
    <dependencies>
        <dependency>
            <groupId>com.saidake</groupId>
            <artifactId>aftersale-config</artifactId>
            <version>${project.version}</version>          自定义模块可以使用 project.version 来限制版本
        </dependency>
       <dependency>
            <groupId>org.springframework.boot</groupId>      组织名，通常是域名倒写
            <artifactId>spring-boot-starter-test</artifactId>      包名
            <version>2.0.5.RELEASE</version>                  版本号
            <optional>true</optional>                         依赖对外隐藏，依赖不会传递
            <type>pom</type>
            <scope>test</scope>                                加载包的范围
                compile (default scope)        Available at compile, test, and runtime phases.
                test              mvn test 时有效，编译和打包时都不会使用这个依赖
                provided       编译和测试时有效，但是运行时不需要该 jar 包，不传递的依赖包， compile 编译 和 打包 的
```

时候被加入进来

runtime 运行时才会依赖，编译时不会依赖。比如，在编译时不需要 JDBC API 的 jar 包，而在运行时才需要 JDBC 驱动包，就可以用 runtime 修饰

system 与 provided 相同，不过被依赖项不会从 maven 仓库获取，而是从本地文件系统拿，需要配合 systemPath 使用

import maven 多模块项目结构中，可以使用 parent 定义父项目，实现从父项目中继承依赖 scope 的 import 属性只能在<depencencyManagement>中使用，表示从其它的 pom 文件中导入 dependency 配置

```
<exclusions>          主动断开依赖的资源，被排除的资源无需指定版本，不会覆盖前方项目的同样的依赖版本
    <exclusion>
        <groupId>org.hamcrest</groupId>
        <artifactId>hamcrest-core</artifactId>
    </exclusion>
</exclusions>
</dependency>
</dependencies>
</dependencyManagement>
```

Build

```
<build>      包括项目的源码目录配置、输出目录配置、插件配置、插件管理配置等
    <filanlName>${artifactId}-${version}</finalName>    最终打包名称   xxapp.jar
    <defaultGoal>install</defaultGoal>                  执行构建时默认的 goal 或 phase，如 jar 或者 package 等
    <directory>${basedir}/target</directory>            构建的结果所在的路径，默认为${basedir}/target 目录
    <plugins>
        <plugin>
        </plugin>
    </plugins>

    <pluginManagement>      子工程默认插件版本和配置（父工程使用）
        <plugin>
            <groupId>com.example</groupId>
            <artifactId>my-maven-plugin</artifactId>
            <version>1.0-SNAPSHOT</version>
            <executions>
                <execution>
                    <phase>compile</phase>          Specific lifecycle
                    <goals>
                        <goal>my-mojo</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <basedir>${project.basedir}</basedir>
            </configuration>
        </plugin>
    </pluginManagement>
```

```
        <resources>      在任意配置文件中加载 pom 文件中定义的属性（例如 jdbc.properties 会被加载 pom 定义的属性）
            <resource>
                <directory>${project.basedir}/src/main/resources</directory>        <!--   扫描的资源文件  -->
                <targetPath>META-INF/resources/</targetPath>                        <!--   将资源文件都放在 target 目录  -->
            </resource>
             <resource>
                <directory> ${project.basedir}/src/main/resources</directory>       <!--       代表所有子项目能被匹配的路径
-->
                <filtering>true</filtering>
                <includes>
                    <include>**/*.properties</include>                    <!-- 手动识别资源  -->
                </includes>
              </resource>
            <testResources>
              <testResource>
                    <directory>${project.basedir}/src/testV resources</directory>
                    <filtering>true</filtering>
               </ testResource>
            </ testResources>
        </ resources>
</build>



<licenses>
        <license>
            <name>The Apache Software License, Version 2.0</name>
            <url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
        </license>
</licenses>
<developers>
        <developer>
            <name>LeTao Zhu</name>
            <email>zltdiablo@163.com</email>
            <organizationUrl>https://github.com/zlt2000</organizationUrl>
            <url>https://blog.csdn.net/zlt2000</url>
        </developer>
</developers>

<!-- nexus 私服配置，项目的仓库配置-->
<repositories>              <!--   maven 会优先采用这个配置，而不会去读 setting.xml 中的配置了，可配置多个仓库  -->
    <repository>
        <id>nexus2</id>         <!-- 区分不同的 mirror 元素，访问指定的仓库  -->
        <name>Nexus Repository</name>
        <url>http://ip:8081/repository/maven-public/</url>
        <releases>
```

```
            <enabled>true</enabled>
        </releases>
        <snapshots>          <!--snapshots 默认是关闭的,需要开启 -->
            <enabled>true</enabled>
        </snapshots>
    </repository>
</repositories>

<distributionManagement>      项目的部署配置（根据当前项目 pom 文件中的 version 判断发布的位置）
        <repository>
            <id>sdk-releases</id>
            <url>http://192.168.22.129:8084/repository/sdk-releases</url>
        </repository>
        <snapshotRepository>
            <id>sdk-snapshots</id>
            <url>http://192.168.22.129:8084/repository/sdk-snapshots</url>
        </snapshotRepository>
</distributionManagement>
```

Commands

mvn

Core

mvn

-f <path> Specifies the path to the POM file or project directory, forcing the use of an alternative POM file (must follow -f with the path).

-e Displays detailed error messages.

-B Runs in batch mode (non-interactive) and disables output color.

-X Enables debug output for more detailed execution information.

-l <log-file> Redirects all build output to the specified log file (disables output color).

-q Runs in quiet mode, displaying only errors.

-v Displays the Maven version.

-h Displays help information.

-D <property=value> Defines a system property.

-P <profile> Activates the specified build profile (e.g., -Ppro_env to load the configuration for the pro_env environment).

-DskipTests

Skips tests during the package phase (compiles the tests but does not run them).

-Dmaven.test.skip=true (Maven 2.2.1)

Skips the test execution during the build process (commonly used with mvn clean package to build the project without running tests).

mvn test

Runs the project's test cases.

The environment variables configured for starting SpringApplication typically won't take effect in JUnit tests or when running mvn test.

Pass environment variables to a junit test:

- @TestPropertySource(properties = "key=value") for Spring tests
- System.setProperty("key", "value") before the test runs
- Configuring the surefire plugin in pom.xml to pass environment variables

-Dtest=UserServiceTest#testUserCreation

Executes a specific test method, in this case, testUserCreation within the UserServiceTest class.

-Dsurefire.log.level=DEBUG

This parameter sets the logging level. You can specify levels like DEBUG, INFO, WARN, or ERROR.

-Dsurefire.printSummary

Set this to true to print a summary of the test results after execution. This can provide a quick overview of what happened during the test run.

mvn dependency:list

Lists all the dependencies resolved for the current project.

mvn dependency:tree

Displays the project's dependency hierarchy. To get an up-to-date view, perform a clean install beforehand to clear outdated dependencies.

mvn dependency:purge-local-repository

Removes local copies of dependencies and forces a re-download from the remote repositories.

mvn dependency:copy-dependencies

Install dependencies without executing any other tasks (like compiling or running tests)

mvn clean

mvn clean package dockerfile:build

Builds the project and generates a Docker image using the Docker plugin in the local Docker environment.

mvn clean package

Cleans the project and builds the package.

mvn package

Packages the project into a distributable format (e.g., JAR or WAR).

mvn site

Generates project documentation based on the Maven site plugin.

mvn compile

Downloads all dependencies and compiles the project's source code.

mvn test-compile

Compiles the project's test source code.

mvn source:jar

Packages the project's source code into a JAR file.

mvn validate

Validates the project structure and configuration.

mvn install

Installs the packaged artifact to the local Maven repository for use by other projects.

A POM module cannot be used as a regular dependency since it is not a JAR;

it should be utilized for dependency management instead.

mvn install:install-file

-Dfile=<source-file-path>

-DgroupId=<group-id>

-DartifactId=<artifact-id>

-Dversion=<version>

-Dpackaging=<packaging>

Installs an external JAR to the local repository manually.

mvn install:install-file

-Dfile=<source-file-path>
-DpomFile=<pom-file-path>
Installs a JAR file along with its associated POM file to the local repository.

Deployment

mvn deploy
Deploys the built project to a remote repository (e.g., a private repository or a repository manager like Nexus).
Deletes the target directory and its contents, effectively cleaning up compiled files and other generated artifacts.

mvn jar
This command is not typically used directly. Use mvn package instead to generate a JAR.

mvn spring-boot:run
Starts a Spring Boot application directly from the command line.

mvn jetty:run
Runs the web application on an embedded Jetty server.

mvn versions:set -DnewVersion=1.1-SNAPSHOT
Sets the project's version to 1.1-SNAPSHOT.

mvn versions:commit
Commits the version changes made using the versions:set command.

mvn help:system
Displays system information and configuration details for Maven.
This command is useful for troubleshooting but does not download dependencies.

Archetype

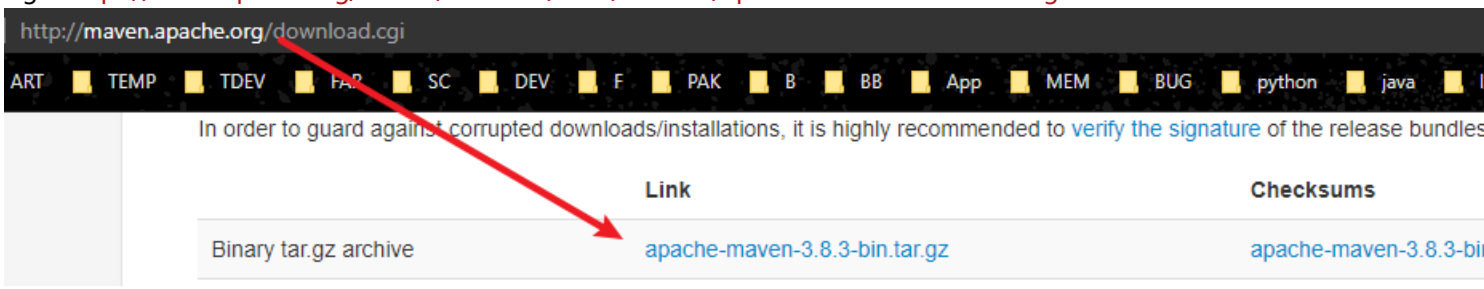
mvn archetype:generate
Creates a new Maven project from an archetype. Common usage:
mvn archetype:generate
-DgroupId=<project-group-id>
-DartifactId=<project-name>
-DarchetypeArtifactId=maven-archetype-quickstart
-DinteractiveMode=false
Creates a basic Java project.
mvn archetype:generate
-DgroupId=com.example
-DartifactId=java-project
-DarchetypeArtifactId=maven-archetype-quickstart
-Dversion=0.0.1-SNAPSHOT
-DinteractiveMode=false
Generates a Java project with specified group ID and artifact ID.
mvn archetype:generate
-DgroupId=com.example
-DartifactId=web-project
-DarchetypeArtifactId=maven-archetype-webapp
-Dversion=0.0.1-SNAPSHOT
-DinteractiveMode=false
Creates a web project with a specified version.

Installation

local 安装：

1. 官网下载二进制包：https://maven.apache.org/download.cgi

wget https://dlcdn.apache.org/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.tar.gz



解压缩： tar xzvf /srv/ftp/apache-maven-3.6.2-bin.tar.gz -C /usr/local/

2. 添加 maven 环境变量

vi /etc/profile

export MAVEN_HOME=/usr/local/apache-maven-3.6.2

export PATH=$PATH:$MAVEN_HOME/bin

source /etc/profile 生效配置

3. 配置本地仓库镜像：vi /usr/local/apache-maven-3.6.2/conf/settings.xml 全局 settiong 定义了当前计算机中 Maven 的公共配置，用户 setting 定义了当前用户的配置

4. 测试成功 mvn -version

yum 安装：

1. wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

2. yum -y install apache-maven

settings.xml

Priority

global configuration: ${maven.home}/conf/settings.xml

user configuration（high priority）: ${user.home}/.m2/settings.xml

settings.xml

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                              https://maven.apache.org/xsd/settings-1.0.0.xsd">

  <!-- The local repository where Maven stores downloaded artifacts.
       By default, Maven uses ~/.m2/repository. You can customize the path if needed. -->
  <localRepository>/path/to/your/local/repo</localRepository>

  <!-- Interactive mode: If set to 'true', Maven may prompt the user for input.
       If set to 'false', it will avoid interactive prompts and continue with default values. -->
  <interactiveMode>true</interactiveMode>

  <!-- Offline mode: If set to 'true', Maven will only use the local repository
       and will not try to download dependencies from remote repositories. -->
  <offline>false</offline>

  <!-- Whether to use the plugin registry file (.m2/plugin-registry.xml).
       Deprecated in Maven 3.x and has no effect in newer versions. -->
  <usePluginRegistry>true</usePluginRegistry>

  <!-- Specifies additional plugin group IDs to search for Maven plugins.
       This allows you to use plugins from these groups without specifying the full group ID. -->
  <pluginGroups>
    <!-- Example plugin group for a custom plugin provider -->
    <pluginGroup>com.example.plugins</pluginGroup>
```

```
    <!-- Another custom plugin group -->
    <pluginGroup>org.thirdparty.plugins</pluginGroup>
</pluginGroups>

<!-- Proxy configuration: Used if Maven needs to access the internet through a proxy server. -->
<proxies>
    <proxy>
        <!-- A unique identifier for this proxy configuration. -->
        <id>example-proxy</id>

        <!-- Whether this proxy is active. -->
        <active>true</active>

        <!-- The protocol for the proxy, e.g., 'http' or 'https'. -->
        <protocol>http</protocol>

        <!-- The hostname or IP address of the proxy server. -->
        <host>proxy.example.com</host>

        <!-- The port number on which the proxy server is listening. -->
        <port>8080</port>

        <!-- Optional: Username for authentication with the proxy server. -->
        <username>proxyuser</username>

        <!-- Optional: Password for authentication with the proxy server. -->
        <password>somepassword</password>

        <!-- Hosts that should be accessed directly, bypassing the proxy.
             Multiple hosts can be separated by a pipe '|' character. -->
        <nonProxyHosts>localhost|127.0.0.1</nonProxyHosts>
    </proxy>
</proxies>

<!-- Server configuration: Used to provide credentials for interacting with remote repositories
     or other services that require authentication (e.g., deploying artifacts). -->
<servers>
    <server>
        <!-- A unique identifier that matches the repository ID in the POM file. -->
        <id>internal-repo</id>

        <!-- Username used for authentication with the repository. -->
        <username>deployuser</username>

        <!-- Password used for authentication with the repository. -->
        <password>deploypassword</password>

        <!-- Optional: Can specify other credentials like private keys if needed. -->
    </server>
    <httpConfiguration>
        <!-- The time in milliseconds to wait for a connection to be established. -->
        <all>
            <connectionTimeout>20000</connectionTimeout> <!-- 20 seconds -->
            <readTimeout>30000</readTimeout> <!-- 30 seconds -->
            <retries>3</retries> <!-- Number of times to retry on failure -->
        </all>
    </httpConfiguration>
</servers>

<!-- Mirrors: Alternative repositories that Maven should use instead of the default repositories.
     This is useful for using a corporate repository or a faster mirror. -->
<mirrors>
    <mirror>
        <!-- A unique identifier for the mirror. -->
        <id>central-mirror</id>

        <!-- Specifies which repositories this mirror is a replacement for.
             'central' means it replaces Maven Central. -->
```

```
          <mirrorOf>central</mirrorOf>
          <!--
                The URL of the mirror repository.
                <mirrorOf>central</mirrorOf>
                    If you set <mirrorOf> to a specific repository ID (e.g., central), Maven will use the mirror
                    for that repository only.
                <mirrorOf>*</mirrorOf>
                    Using * means that the mirror will be used for all repositories.
                    If no exact match is found, Maven will look for a mirror with <mirrorOf>*</mirrorOf>, which
                    is configured to match all repositories. This is a catch-all rule.
                    If multiple mirrors have <mirrorOf>*</mirrorOf>, the first defined mirror in the settings.xml
                    file will be used.
                <mirrorOf>central,internal-repo</mirrorOf>
                    You can specify a list of repository IDs separated by commas. The mirror will apply to any
                    repositories that match one of the specified IDs.
                    If a mirror is configured with <mirrorOf>central,internal-repo</mirrorOf>, it will match
                    either central or internal-repo.
                    If multiple such matches exist, the first one in the configuration order is chosen.
                <mirrorOf>*,!internal-repo</mirrorOf>
                    If you want to exclude a specific repository, you can prefix its name with !. This means
                    the mirror will be used for all repositories except the specified one.
                <mirrorOf>external:*</mirrorOf>
                    You can use patterns like external:* to match all external repositories (i.e., repositories
                    not hosted locally).
          -->

          <url>https://my.company.repo/maven2</url>

          <!-- The repository layout; 'default' is the usual choice. -->
          <layout>default</layout>
       </mirror>
    </mirrors>

    <!-- Profiles: Custom configurations that can be activated under certain conditions.
         Profiles can modify settings like properties, repositories, and plugin configurations. -->
    <profiles>
       <profile>
          <!-- A unique identifier for this profile. Maven will search that custom repository for artifacts if
they are not found in the central repository or any other defined repositories. -->
          <id>development</id>
          <repositories> <!-- List of remote repositories to use under this profile -->
             <repository>
                <id>central</id>
                <!--
                    The identifier for the repository, used for matching purposes
                    If you do not define the central repository in either the <repositories> or
                   <pluginRepositories> sections of your settings.xml or pom.xml,
                    Maven will not be able to find artifacts from the Maven Central repository, and will fail
                   to find it and produce an error.
                -->
                <name>Maven Central Repository</name> <!-- Human-readable name for the repository -->
                <url>https://repo2.maven.org/maven2</url> <!-- The URL for accessing the repository -->

                <!-- Configure settings for release artifacts -->
                <releases>
                   <enabled>true</enabled> <!-- Indicates whether this repository is enabled for release
artifacts, The default value is true. -->
                </releases>

                <!-- Configure settings for snapshot artifacts -->
                <snapshots>
                   <enabled>true</enabled> <!-- Indicates whether this repository is enabled for snapshot
artifacts, The default value is true. -->
                   <updatePolicy>always</updatePolicy>
                      <!-- Update policy for checking new versions, Optional, default is daily
                           - "always" means to check for updates every time
                           - "never" means to disable the check for updates,
                           - "daily" means to check once per day,
```

```
                        - "interval:X" means to check every X minutes
                    -->
                </snapshots>
            </repository>
        </repositories>

        <!-- List of plugin repositories -->
        <pluginRepositories>
            <pluginRepository> <!-- A specific plugin repository -->
                <id>central</id> <!-- Unique identifier for the repository -->
                <url>https://repo.maven.apache.org/maven2</url> <!-- URL of the repository -->

                <releases> <!-- Configuration for release artifacts -->
                    <enabled>true</enabled> <!-- Indicates that this repository can serve release artifacts -->
                </releases>

                <snapshots> <!-- Configuration for snapshot artifacts -->
                    <enabled>true</enabled> <!-- Indicates that this repository can serve snapshot artifacts
-->
                    <updatePolicy>always</updatePolicy> <!-- Update policy for snapshots -->
                </snapshots>
            </pluginRepository>
        </pluginRepositories>

        <!-- Properties that can be used to customize builds or provide environment-specific values. -->
        <properties>
            <env>dev</env>
        </properties>
    </profile>
  </profiles>

  <!-- Active profiles: Profiles that should be activated by default during the build process.
       You can activate multiple profiles by specifying them here. -->
  <activeProfiles>
      <activeProfile>development</activeProfile>
  </activeProfiles>
</settings>
```

<localRepository>/usr/data/maven/local_repository</localRepository> The path to the local repository, its default value is "~/.m2/repository"

<interactiveMode>true</interactiveMode> 是否需要和用户交互以获得输入，默认为 true。

<usePluginRegistry>false</usePluginRegistry> 是否需要让 Maven 使用文件~/.m2/plugin-registry.xml 来管理插件版本，默认为 false。

<offline>false</offline> 是否需要在离线模式下运行，默认为 false。

<pluginGroups> 当插件的组织 id（groupId）没有显式提供时，提供搜寻插件组织 Id（groupId）的列表。默认情况下该列表包含了 org.apache.maven.plugins 和 org.codehaus.mojo。

  <pluginGroup>org.codehaus.mojo</pluginGroup> plugin 的组织 Id（groupId）

</pluginGroups>

<mirrors> When downloading artifacts from a remote repository, The mirror is used to replace the repository.

  priority: repository（setting.xml） < repository（pom.xml） < mirror（setting.xml）

  <mirror> 指定仓库的下载镜像

    <id>alimaven</id> Mirror id, It will also match server.id to use authentication configuration.

    <mirrorOf>*</mirrorOf> 对哪种仓库进行镜像，简单说就是替代哪个仓库

      * 匹配所有仓库请求，即将所有的仓库请求都转到该镜像上，其他仓库配置都会失效

      central,repo1,repo2 将仓库 central，repo1，repo2 的请求转到该镜像上，使用逗号分隔多个远程仓库， mirrorOf 重复时，默认取第一个进行转发，后续 mirror 无效

```
        *,!repo1                匹配所有仓库请求，repo1 除外，使 r 用感叹号将仓库从匹配中排除，互斥仓库会同时生效，找不到会尝试其他仓库
    <name>aliyun maven</name>       镜像名称
    <url>http://maven.aliyun.com/nexus/content/repositories/central/</url>      镜像 URL
</mirror>

<mirror>        <!-- 自定义的私服 -->
    <id>nexus-sdk</id>
    <mirrorOf>*</mirrorOf>
    <url>http://192.168.22.129:8084/repository/maven-public</url>
</mirror>

<!-- 阿里云仓库 -->
<mirror>
  <id>alimaven</id>
  <mirrorOf>central</mirrorOf>
  <name>aliyun-maven</name>
  <url>http://maven.aliyun.com/nexus/content/repositories/central/</url>
</mirror>

<!-- 中央仓库 1 -->
<mirror>
  <id>repo1</id>
  <mirrorOf>central</mirrorOf>
  <name>Human Readable Name for this Mirror.</name>
  <url>https://repo1.maven.org/maven2/</url>
</mirror>

<!-- 中央仓库 2 -->
<mirror>
  <id>repo2</id>
  <mirrorOf>central</mirrorOf>
  <name>Human Readable Name for this Mirror.</name>
  <url>https://repo2.maven.org/maven2/</url>
</mirror>

<!-- 腾讯云 -->
<mirror>
  <id>nexus-tencentyun</id>
  <mirrorOf>central</mirrorOf>
  <name>Nexus tencentyun</name>
  <url>http://mirrors.cloud.tencent.com/nexus/repository/maven-public/</url>
</mirror>

<!--华为云 -->
<mirror>
  <id>huaweicloud</id>
  <mirrorOf>central</mirrorOf>
```

```
        <url>https://mirrors.huaweicloud.com/repository/maven/</url>
    </mirror>
</mirrors>
<servers>       Authorization configuration information required for remote mirror server access.
        <server>
            <id>alimaven</id>
            <username>admin</username>
            <password>admin</password>
            <privateKey>${usr.home}/.ssh/id_dsa</privateKey>     鉴权时使用的私钥位置。和前两个元素类似，私钥位置和私钥密码指定了一个私钥的路径
                                                                 (默认是${user.home}/.ssh/id_dsa）以及如果需要的话，一个密语。
                                                                 将来 passphrase 和 password 元素可能会被提取到外部，但目前它们必须在 settings.xml 文件以纯文本的形式声明。
            <passphrase>some_passphrase</passphrase>             鉴权时使用的私钥密码。
            <filePermissions>664</filePermissions>               文件被创建时的权限。如果在部署的时候会创建一个仓库文件或者目录，这时候就可以使用权限（permission）。
                                                                 这两个元素合法的值是一个三位数字，其对应了 unix 文件系统的权限，如 664，或者 775。
            <directoryPermissions>775</directoryPermissions>     目录被创建时的权限
        </server>
        <server>
            <id>sdk-snapshots</id>
            <username>admin</username>
            <password>admin</password>
        </server>
</servers>


<profiles>          多仓库配置，根据环境参数来调整构建配置的列表。
    <profile>
      <id>repo2</id>
      <repositories>            远程仓库列表
        <repository>
          <id>repo2</id>
          <url>https://repo2.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>          true 或者 false 表示该仓库是否为下载某种类型构件（发布版，快照版）开启。
          </releases>
          <snapshots>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
          </snapshots>
        </repository>
      </repositories>

      <pluginRepositories>        插件仓库列表
        <repository>
```

```
            <id> alimaven </id>
            <url>https://repo2.maven.org/maven2</url>
            <releases>
              <enabled>true</enabled>
            </releases>
            <snapshots>
              <enabled>true</enabled>
              <updatePolicy>always</updatePolicy>
            </snapshots>
          </repository>
        </pluginRepositories>
        <activation>              自动触发 profile 的条件逻辑
          <activeByDefault>true</activeByDefault>       activate by default
          <jdk>1.8</jdk>                                jdk detection
          <os>                                          system dectection
              <name>Windows XP</name>
              <family>Windows</family>
              <arch>x86</arch>
              <version>5.1.2600</version>
          </os>
          <property>                                    pom properties dectection
              <name>mavenVersion</name>
              <value>2.0.3</value>
          </property>
          <file>                                        check if the file exists
              <exists>/path/to/active_on_exists</exists>
              <missing>/path/to/active_on_missing</missing>
          </file>
        </activation>
        <properties>              扩展属性列表
          <maven.compiler.source>1.8</maven.compiler.source>
          <maven.compiler.target>1.8</maven.compiler.target>
          <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>
        </properties>

    </profile>
</profiles>

<proxies>       代理元素包含配置代理时需要的信息
    <proxy>
      <id>myproxy</id>          代理的唯一定义符，用来区分不同的代理元素。
      <active>true</active>          该代理是否是激活的那个。true 则激活代理。当我们声明了一组代理，而某个时候只需要激活一个代理的时候， 该元素就可以派上用处。
      <protocol>http</protocol>     代理的协议。 协议://主机名:端口，分隔成离散的元素以方便配置。
      <host>proxy.somewhere.com</host>          代理的主机名。协议://主机名:端口，分隔成离散的元素以方便配置。
      <port>8080</port>                          代理的端口。协议://主机名:端口，分隔成离散的元素以方便配置。
      <username>proxyuser</username>             代理的用户名，用户名和密码表示代理服务器认证的登录名和密码。
      <password>somepassword</password>               代理的密码，用户名和密码表示代理服务器认证的登录名和密码。
```

<nonProxyHosts>*.google.com|ibiblio.org</nonProxyHosts>　　不该被代理的主机名列表。该列表的分隔符由代理服务器指定；例子中使用了竖线分隔符，　使用逗号分隔也很常见。

</proxy>

</proxies>

Server

nexus 仓库安装

Nexus 安装网址：　https://help.sonatype.com/repomanager3/product-information/download

vim　./bin/nexus.vmoptions　　虚拟机选项配置文件

vim ./etc/nexus-default.properties　　启动选项配置

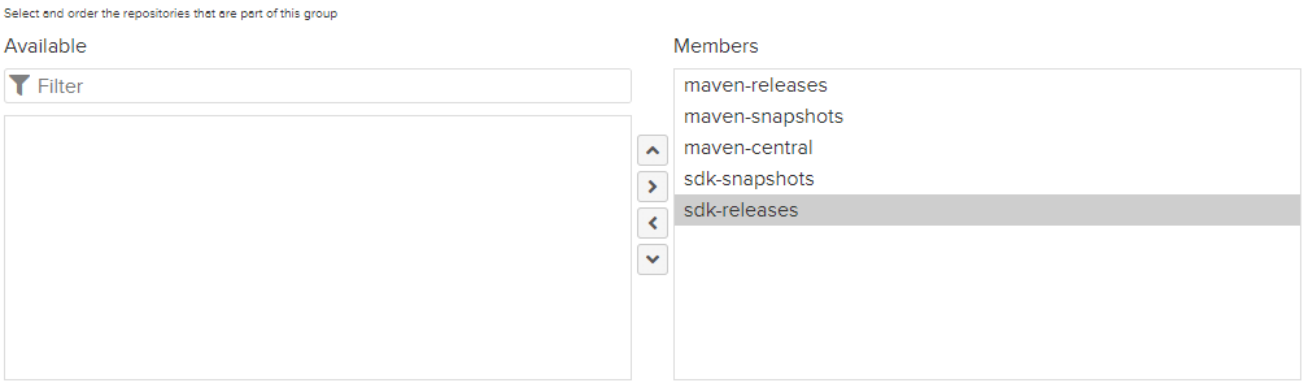
nexus {start|stop|run|run-redirect|status|restart|force-reload}　　命令（启动时间会比较慢 要 1 分钟左右后才能访问。）

cat cat /usr/local/sonatype-work/nexus3/admin.password　　查看 admin 默认用户登录密码

nexus 仓库配置

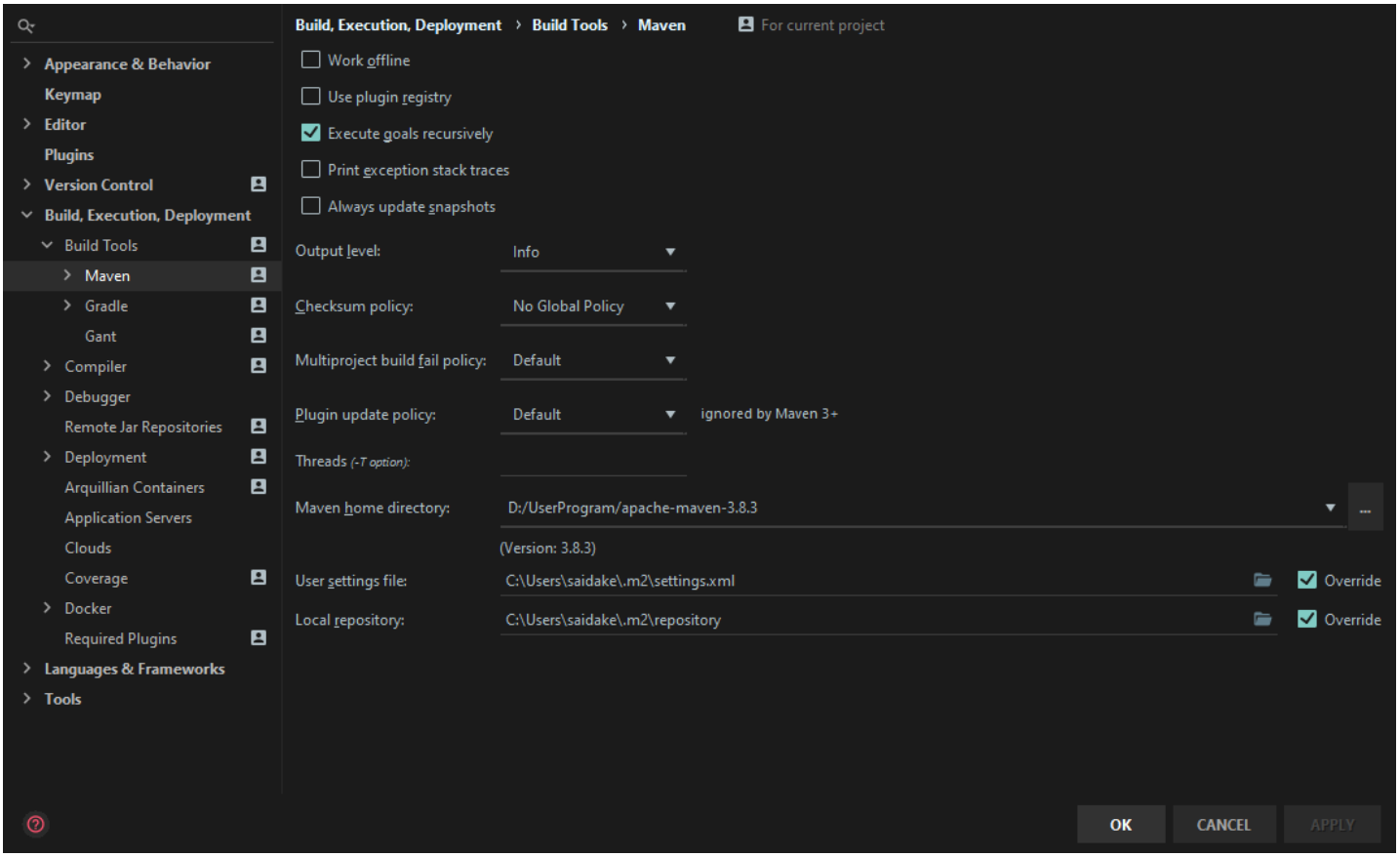
创建仓库（sdk-snapshots，sdk-release）：

移动到主仓库中 maven-public（maven 只用配置 maven-public 仓库就行了）



项目部署使用

idea 配置使用本地 maven：



项目添加私服：项目 pom.xml 添加部署配置：　　<dirtributionManagement >

　　修改 maven 的 setting.xml 仓库配置：　<servers>　<mirrors>

gadle

Configuration

Compatibility

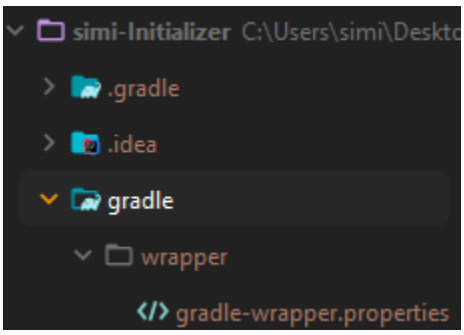
Java	11
gradle	5.0
org.jetbrains.intellij	0.7.3
intellij	2021.3.1

Gradle user home: C:\Users\simi\.gradle

~/.gradle/gradle.properties

intellijPublishToken=perm:c2FpZGFrZQ==.OTItOTUxMA==.xxxxxxxxxxxxxxxx　　# Add your plugin token

gradle-wrapper.properties



gradle/wrapper/gradle-wrapper.properties >>　　Create if it does not exist

distributionBase=GRADLE_USER_HOME

distributionPath=wrapper/dists

distributionUrl=https\://services.gradle.org/distributions/gradle-5.0-bin.zip

#distributionUrl=https\://mirrors.cloud.tencent.com/gradle/gradle-5.0-bin.zip

zipStoreBase=GRADLE_USER_HOME

zipStorePath=wrapper/dists

setting.gradle

```
plugins {
    id "io.spring.develocity.conventions" version "0.0.22"
```

A plugin that provides Spring-specific conventions.

```
    id "org.gradle.toolchains.foojay-resolver-convention" version "0.9.0"
```

A plugin related to toolchain resolution and handling.

```
}
```

```
rootProject.name='simi'
```

This sets the root project name for the Gradle build to "simi".

```
include 'sagan-site', 'sagan-client'
```

Include child gradle projects in the root directory.

```
rootProject.children.each { project ->
    project.buildFileName = "${project.name}.gradle"
}
```

This block dynamically renames the build.gradle file for each subproject based on the project name.

This helps in having separate gradle build files per subproject, following a naming convention (project-name.gradle).

```
settings.gradle.projectsLoaded {
    develocity {
        buildScan {
            File buildDir = settings.gradle.rootProject
                    .getLayout().getBuildDirectory().getAsFile().get()
            buildDir.mkdirs()
            new File(buildDir, "build-scan-uri.txt").text = "(build scan not generated)"
            buildScanPublished { scan ->
                if (buildDir.exists()) {
                    new File(buildDir, "build-scan-uri.txt").text = "${scan.buildScanUri}\n"
                }
            }
        }
    }
}
```

This hook executes when all projects are loaded.

It configures the develocity plugin to generate a build scan and writes the URI of the build scan to a file called build-scan-uri.txt in the build directory.

build.gradle

Properties

```
group = 'org.saidake'
```

Defines the group identifier for the project, usually structured like a domain name

```
version = '1.0-SNAPSHOT'
```

Specifies the current version of the project

```
archivesBaseName = 'myapp-archive'
```

Sets the base name for the generated archives (e.g., JAR files).

```
sourceCompatibility = '17'
targetCompatibility = '17'
```

Defines the Java version compatibility for compiling source and target code.

```
repositories
dependencies
```

Specifies the external libraries or projects that the current project depends on. These can be grouped into configurations like implementation, testImplementation, and runtimeOnly.

Dependencies

```
allprojects {
```
allprojects block

This applies to both the parent project and all child projects.

subprojects block

You can also use subprojects { ... } to apply dependencies only to the child projects, excluding the parent project itself.

Unlike dependencies and configurations that can be shared using the allprojects or subprojects blocks, plugins must be explicitly applied to each project (parent or child) because they directly affect the build process by adding tasks and altering project configurations.

```
    repositories {
       mavenCentral()
    }
    dependencies {
       implementation 'org.springframework.boot:spring-boot-starter-webflux'
       implementation 'org.springframework.boot:spring-boot-starter-data-mongodb-reactive'
    }
}

plugins {
```
Used to apply Gradle plugins to the project, such as java, application, or external plugins like id 'org.springframework.boot'.

```
    id 'java'
    id 'org.jetbrains.intellij' version '1.1'
    id "io.spring.dependency-management" version "1.1.3"
}

repositories {
```
Declares the locations (repositories) where Gradle should look for dependencies.

Common options are mavenCentral(), jcenter(), or custom repositories.

```
    mavenCentral()
}
dependencies {
```
Specifies the external libraries or projects that the current project depends on.

These can be grouped into configurations like implementation, testImplementation, and runtimeOnly.

```
    implementation ('org.springframework.boot:spring-boot-starter-webflux') {
       exclude group: 'org.springframework.boot', module: 'spring-boot-starter-logging' // Exclude the default
logging
    }

    implementation platform("org.springframework.boot:spring-boot-dependencies:3.1.1")
```
Import a BOM as a platform for dependency management

```
    compileOnly 'groupId:artifactId:version'                          Add maven denpendencies in compilation.
    runtimeOnly 'org.apache.commons:commons-math3:3.6.1'              Add maven denpendencies in runtime.
    annotationProcessor 'org.projectlombok:lombok:1.18.26'            Lombok annotation processor
    implementation 'com.google.guava:guava:31.1-jre'
    testImplementation 'org.junit.jupiter:junit-jupiter:5.9.1'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.7.2'

}
```

Task

```
task hello {
```
Create custom tasks.

```
    doFirst {
       println 'This runs first.'
```

```
    }
    doLast {
        println 'Hello, World!'
    }
}

project.tasks.create('myTask') {
```

Create custom tasks programmatically.

project.tasks.create(String name, Class<T> type = null, Closure configureClosure = null)

name:	The name of the task.
type (optional):	The type of the task (e.g., Copy, Exec, or a custom class).
configureClosure (optional):	A closure to configure the task's behavior.

```
    doLast {
        println 'Hello from myTask!'
    }
}
project.tasks.create("versionInfo", {
    def buildNumber = System.getenv("BUILD_NUMBER") ?: 0
    project.ext.releaseVersion = version + '.' + buildNumber
})



tasks.named('jar') {
```

This method is used when you want to configure a specific task by its name.

It helps to avoid creating the task if it hasn't been defined yet, ensuring that configuration is applied only to an existing or to-be-defined task.

```
    enabled = false
    archiveBaseName.set('my-custom-jar')

    useJUnitPlatform()
}

tasks.withType(JavaCompile) {
```

This method is used to configure all tasks of a specific type.

It allows you to apply a configuration to every task of that type, which is useful when there are multiple tasks of the same kind (e.g., all JavaCompile tasks).

```
    options.encoding = 'UTF-8'  // Use UTF-8 when compiling JAVA files
    options.compilerArgs += '-Xlint:unchecked'
}
tasks.withType(Javadoc) {
    options.encoding = "UTF-8"
}
apply from: file('gradle/check.gradle')
```

This statement is used to apply an external Gradle script into the current build script.

The from keyword specifies the path to the external Gradle script file that you want to apply.

Suppose you have a gradle/check.gradle file that contains additional configuration or tasks for code quality checks.

```
// gradle/check.gradle
task checkStyle(type: Checkstyle) {
    toolVersion = '8.45'
    source = sourceSets.main.allSource
    configFile = file('config/checkstyle/checkstyle.xml')
}
```

This setup applies the tasks and configurations defined in check.gradle to your build. You can then run these tasks using:

```
gradle checkStyle
gradle lint
```

```
sourceSets {
```
Configures source code directories for different tasks.

```
    main {
        java {
            srcDirs = ['src/main/java']
        }
        resources {
            srcDirs = ['src/main/resources']
        }
    }
}
```
Build

```
build {
```
This refers to Gradle's default lifecycle task called build.

The build task typically depends on several other tasks such as assemble, test, and check.

These tasks ensure that the project is properly compiled, tested, and assembled before the final artifact (like a JAR) is produced.

```
    dependsOn clean, jar
}
```
Global

```
ext {
```
Define extra properties that can be accessed throughout the build script.

These extra properties are dynamic and can be added to the project object, making them available for use in various parts of the build configuration.

```
    springVersion = "3.1.0.RELEASE"
    emailNotification = "build@master.org"
}
```

```
configurations {
```
Configure dependency configurations in Gradle.

Configurations are named sets of dependencies and are used by different tasks to resolve and include dependencies.

```
    implementation.exclude group: "org.springframework.boot", module: "spring-boot-starter-logging"
```
The exclude method is used to remove specific transitive dependencies from the implementation configuration.

It prevents these dependencies from being included in the classpath.

```
    implementation.exclude group: "org.springframework.boot", module: "spring-boot-starter-tomcat"

    all {
            exclude group: 'org.springframework.boot', module: 'spring-boot-starter-logging'
    }
```
Globally Exclude Dependencies for All Configurations

```
}
```

```
application {
```
Configure settings specific to the application plugin.

This plugin is part of the Gradle Java plugin and is used for packaging a Java application, including generating an executable JAR file that can be run directly.

```
    mainClass = 'com.example.myapp.Main'
```
Specifies the fully qualified name of the main class that contains the main method to be executed when the JAR file is run.

```
}
```

Spring Dependency Management Plugin

```
plugins {
    id 'io.spring.dependency-management' version '1.1.3' // Dependency Management Plugin
}
dependencyManagement {
```

Import a BOM from Spring Boot or any other dependency

Optionally manage specific versions of other dependencies

```
    imports {
        mavenBom "org.springframework.boot:spring-boot-dependencies:3.1.1"
    }

    dependencies {
        dependency 'com.google.guava:guava:31.1-jre'
        dependency 'org.apache.commons:commons-lang3:3.12.0'
    }
}
```

Java Plugin

```
jar {
```

The jar block is associated with the Java Plugin in Gradle.

This plugin adds support for building Java projects and includes several tasks related to compiling Java source code, running tests, and packaging applications.

```
    sourceCompatibility = '11'
    targetCompatibility = '11'

    enabled false
```

This disables the creation of a standard ("plain") JAR file.

In Spring Boot 2.5.x or later, Spring Boot projects typically create both a "fat" executable JAR (with dependencies bundled) and a "plain" JAR (without dependencies).

Setting enabled false prevents the creation of the plain JAR, leaving only the executable JAR.

spring 2.5.+; yes, disable of creating the `plain` jar

```
    dependsOn versionInfo
```

Specifies that the jar task depends on the versionInfo task.

This means the versionInfo task must be executed before the jar task.

This could be useful if versionInfo prepares version-related information (like creating a version file).

```
    archiveBaseName.set(artifactId)
```

Sets the base name of the JAR file to artifactId.

This typically refers to the project's identifier (defined elsewhere, possibly in build.gradle or a project property).

It overrides the default JAR file name.

```
    archiveVersion.set(releaseVersion)
```

Sets the version of the JAR file to releaseVersion.

This might be a custom property or variable defined earlier in the build script. It overrides the default version (usually based on version).

```
    manifest {
```

This specifies custom attributes to be included in the JAR's MANIFEST.MF file.

```
        attributes(
            'Implementation-Title': 'MyApp',
            'Implementation-Version': version
        )
    }
```

}

processResources {

The processResources task in Gradle is part of the standard Java plugin tasks and is responsible for processing and copying resource files from the source directories to the output directory.
This task typically includes operations such as filtering, expanding, and copying resources.

`expand(project.properties)`

If you have a config.properties file with placeholders like ${version}, and you want these placeholders to be replaced with actual values from the project properties,
you would use the expand method.

expand

This method is used for property-based filtering of resource files.
It replaces placeholders in your resource files with the corresponding property values.

project.properties

This provides a map of properties available in the project.
These properties can be used as placeholders in your resource files.

`dependsOn versionInfo`

This keyword specifies that the processResources task depends on another task, in this case, versionInfo.

}

Spring Boot plugin

bootRun {

The bootRun task in Gradle is part of the Spring Boot plugin and is used to run a Spring Boot application directly from the Gradle build.

`jvmArgs = ['-Dspring.profiles.active=local', '-Dlog4j.configurationFile=log4j2-local.xml']`

This property allows you to pass JVM arguments to the Java Virtual Machine when running the Spring Boot application.

}

Reactor Debug

To enable Reactor debug mode in IntelliJ IDEA, you can do the following:

1. Press Ctrl Alt 0S to open Settings
2. Select Languages & Frameworks then Reactive Streams
3. Check the Enable Reactor Debug option
4. Select Hooks as the debug initialization method

Command

gradlew

./gradlew :my-subject:clean

Execute subproject commands

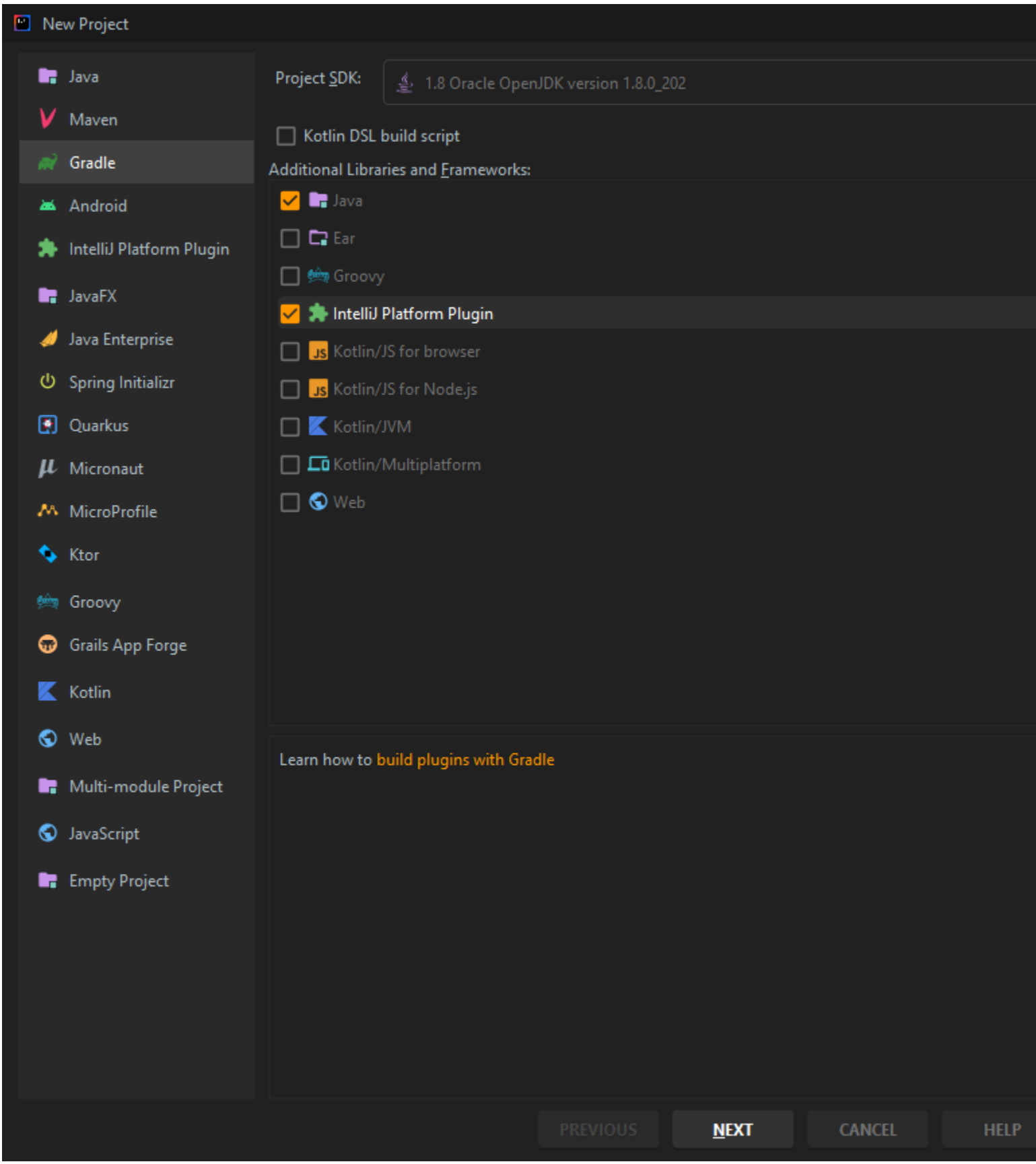
./gradlew dependencies

This command will display the complete dependency tree for all configurations (like implementation, testImplementation, etc.).

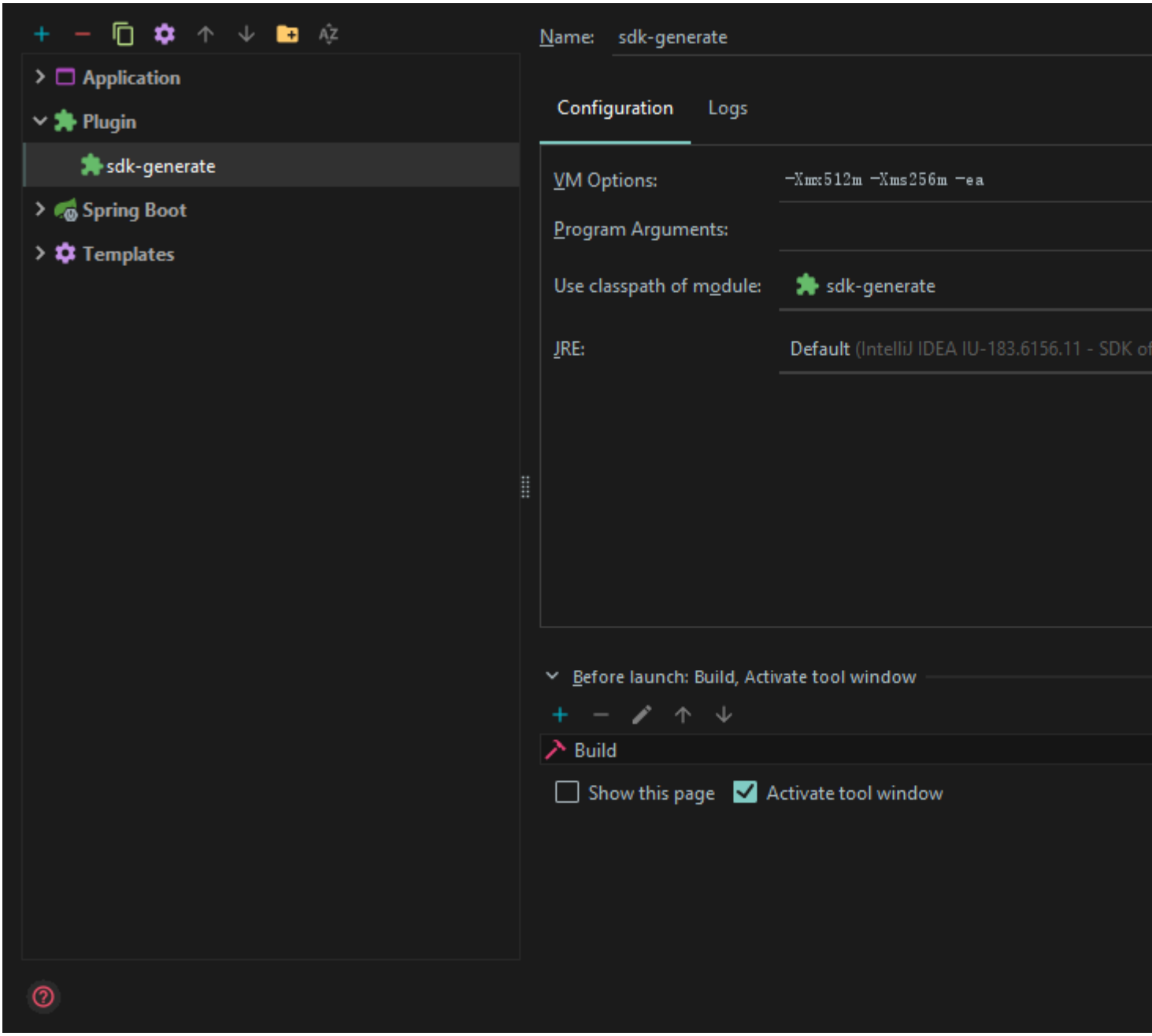
--configuration implementation

If you want to check the dependency tree for a specific configuration (for example, implementation or testImplementation), you can specify the configuration like this

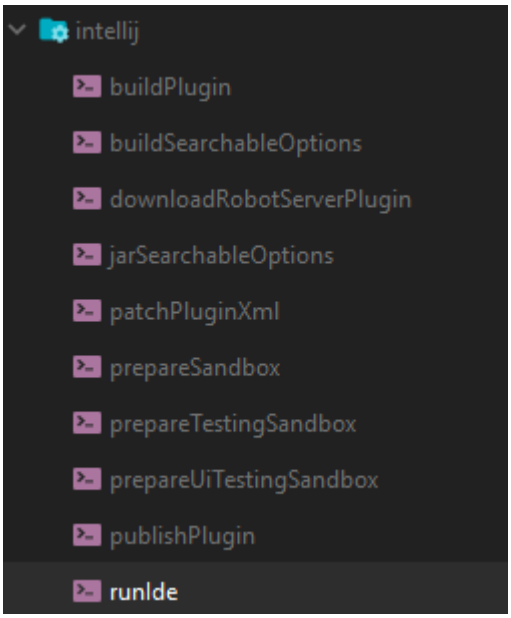
Create Gradle Project



Debug Configuration



Debug plugin with intelij ide



Publish plugin

META-INF/plugin.xml >>

```
<idea-plugin>
  <id>com.saidake.plugin.generate</id>        插件唯一 id
  <name>saidake-generate</name>               插件名称
  <version>1.2.0</version>                    版本号
  <vendor email="saidake@qq.com" url="https://wx.mail.qq.com/login/loginpage">YourCompany</vendor>     公司信息

  <description><![CDATA[                                  插件描述
      Enter short description for your plugin here.<br>
      <em>most HTML tags may be used</em>
    ]]></description>

  <change-notes><![CDATA[
      Add change notes here.<br>
      <em>most HTML tags may be used</em>
    ]]>
  </change-notes>

  <!-- please see http://www.jetbrains.org/intellij/sdk/docs/basics/getting_started/build_number_ranges.html for description
-->
  <idea-version since-build="173.0" until-build="2019.2.*"/>              支持的 idea 版本号

  <!-- please see http://www.jetbrains.org/intellij/sdk/docs/basics/getting_started/plugin_compatibility.html
       on how to target different products -->
  <!-- uncomment to enable plugin in all products
  <depends>com.intellij.modules.lang</depends>
  -->
  <depends>org.jetbrains.idea.maven</depends>             三方依赖
```

```
<resource-bundle>test.properties</resource-bundle>      绑定 resoruces 里的使用属性文件


<extensions defaultExtensionNs="com.intellij">                扩展内容
  <toolWindow id="notewindow" factoryClass="com.itheima.markbook.window.NoteListWindowFactory"        工具窗口
              anchor="right" icon="/img/logo.svg">
  </toolWindow>
  <notificationGroup id="Smp Notification "   displayType="BALLOON"/>          NotificationGroup

</extensions>

<application-components>                                加载应用级别组件
  <component>
    <implementation-class></implementation-class>
  </component>
</application-components>


<actions>
    <!-- Add your actions here -->
    <action id="InitProjectId" class="com.saidake.plugin.init.InitProject" text="smp-init" icon="MyIcons.Action">
        <add-to-group group-id="ToolsMenu" anchor="last"/>
        <keyboard-shortcut keymap="$default" first-keystroke="ctrl F12"/>
    </action>
</actions>


</idea-plugin>
```

Groovy

Variables and Data Types

```
String a = "Groovy"      def a = "Groovy"
    String
def a = 25
    Integer
def a = 19.99
    BigDecimal
double version = 4.0

boolean isAwesome = true
```

Variables (Optional Typing)

```
def name = "Groovy"
String message = "Hello, ${name}!"
println message
```

Lists and Maps

```
def list = [1, 2, 3]
```

```
def map = [key: "value", age: 25]
println list[0]  // Output: 1
println map['key']  // Output: value
```

Closures (Lambda-like Functions)

```
def greet = { name -> println "Hello, $name!" }
greet("Groovy")
```

Class and Method

```
class Person {
    String name
    int age
    void introduce() {
        println "My name is $name, and I am $age years old."
    }
}
def person = new Person(name: "Alice", age: 30)
person.introduce()
```

Config / Deployment Environment

jenkins

安装配置

官网 yum 安装方式：https://www.jenkins.io/doc/book/installing/linux/ Red Hat / CentOS LTS 版本

无法连接官网仓库使用手动：

https://mirrors.jenkins-ci.org/redhat/ 从镜像库中选择一个 rpm 版本

rpm -ivh https://mirrors.jenkins-ci.org/redhat/jenkins-2.350-1.1.noarch.rpm 安装

配置服务文件：vi /lib/systemd/system/jenkins.service（新版本各种配置只用修改这个就行了，优先级最高，老版本修改 jenkins 配置文件：/etc/sysconfig/jenkins）

JAVA_HOME=/usr/local/jdk1.8.0_131

配置执行用户

服务端口，

JENKINS_HOME 安装目录

添加一个 jdk8 环境候选项：vim /etc/init.d/jenkins

```
candidates="
/etc/alternatives/java
/usr/lib/jvm/java-1.8.0/bin/java
/usr/lib/jvm/jre-1.8.0/bin/java
/usr/lib/jvm/java-11.0/bin/java
/usr/lib/jvm/jre-11.0/bin/java
/usr/lib/jvm/java-11-openjdk-amd64
/usr/bin/java
/usr/local/jdk_11.0.1/bin/java
"
```

配置工作目录 /var/lib/jenkins/conf.xml

<workspaceDir>${JENKINS_HOME}/workspace/${ITEM_FULL_NAME}</workspaceDir>

启动服务 systemctl start jenkins

查看日志 /var/log/jenkins/jenkins.log

查看首页：http://192.168.22.133:48123 启动比较慢，需要等待几分钟

全局工具配置（Global Tool Configuration）：

配置工具执行或 HOME 目录（启动后配置） jdk-tool，maven-tool，git-tool // HOME 只用配 安装目录 【配完工具和 git 仓库地址，测一下回调，项目框架就成型了】

启动后配置：

配置插件镜像：Advanced => Update Site = https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json

配置环境变量（启动后配置） JAVA_HOME MAVEN_HOME PATH+EXTRA（$MAVEN_HOME/bin）

ERROR：AWT is not properly configured on this server. Perhaps you need to run your container with "-Djava.awt.headless=true"?

一般 os 安装的是 64 位的，所以缺省是装 libgcc 这个包。但 java 一般还是会用 32 位的包，因此就会存在上面的问题（系统提供的 libgcc_s.so.1 是 64 位的，不是 java 启动需要的 32 位的），安装一个 32 位的就好了。

yum -y install fontconfig

sudo yum install libgcc.i686 --setopt=protected_multilib=false

安装插件

创建多分支项目，添加 git 地址，jenkins 添加 hooks 之后，提交代码后会自动在集成服务器上拉取代码

多分分支流水线项目 Multibranch Scan Webhook Trigger：

1. 配置 多分支： Configure System -->添加凭证， Property strategy 接收不同变量
2. 多分支项目配置 hook： Scan Multibranch Pipeline Triggers：Scan by webhook ，添加 token 【保存时 git 测试：查看日志，git 连接是否正常】【手动回调测试：需要手动 git 提交一次，再查看日志，Jenkinsfile 是否被执行】

多分分支流水线项目 SonarQube Scanner：

1. 全局添加 sonarqube 服务
2. 全局添加 sonar-scanner 工具
3. 本地项目 Jenkinsfile 执行 sonar-scanner // sh "${scannerHome}/bin/sonar-scanner"

邮箱通知 Email Extension，Groovy Postbuild：

1. 全局配置发件人收件人邮箱：Configure System --> Extended E-mail Notification
2. 全局配置发件人邮箱 Configure System --> Jenkins Location
3. 发邮件测试 Test configuration by sending test e-mail

权限分配 Role-based-Authorization Strategy：

1. 配置全局安全策略 Configure Global Security --> 切换授权策略为 Role-Based Strategy
2. 管理角色权限和用户授予角色 Manage and Assign Roles --> Global roles 全局登录角色 Project roles 针对项目角色 Slave roles 节点角色，jenkins 主从机制
3. 角色访问项目匹配，用户能够访问 Pattern 正则匹配到的项目

凭证管理 Credentials Binding：

1. 凭证：点击全局，新建凭证 （Secret file 密钥文件 Secret text 密钥文本 Certificate PKCS 证书导入）
2. 点击项目--配置，源码管理添加 git 地址，并选择创建的凭证拉取代码（http）
3. 创建项目 Freestyle project
4. Build Now ==> /var/lib/jenkins/workspace/itcast01 构建项目到目录

钩子触发器 Generic Webhook Trigger：

1. build Trigger 添加分支 ref 筛选判断（$.ref：Exporession 匹配 post 请求 json 数据内的 ref 成员）
2. 在 git 上添加生成的钩子函数
3. 指定哪个分支触发编译 Post content parameters ---> Variable = ref Expression=$.ref Value filter = $(refs/heads/qa)

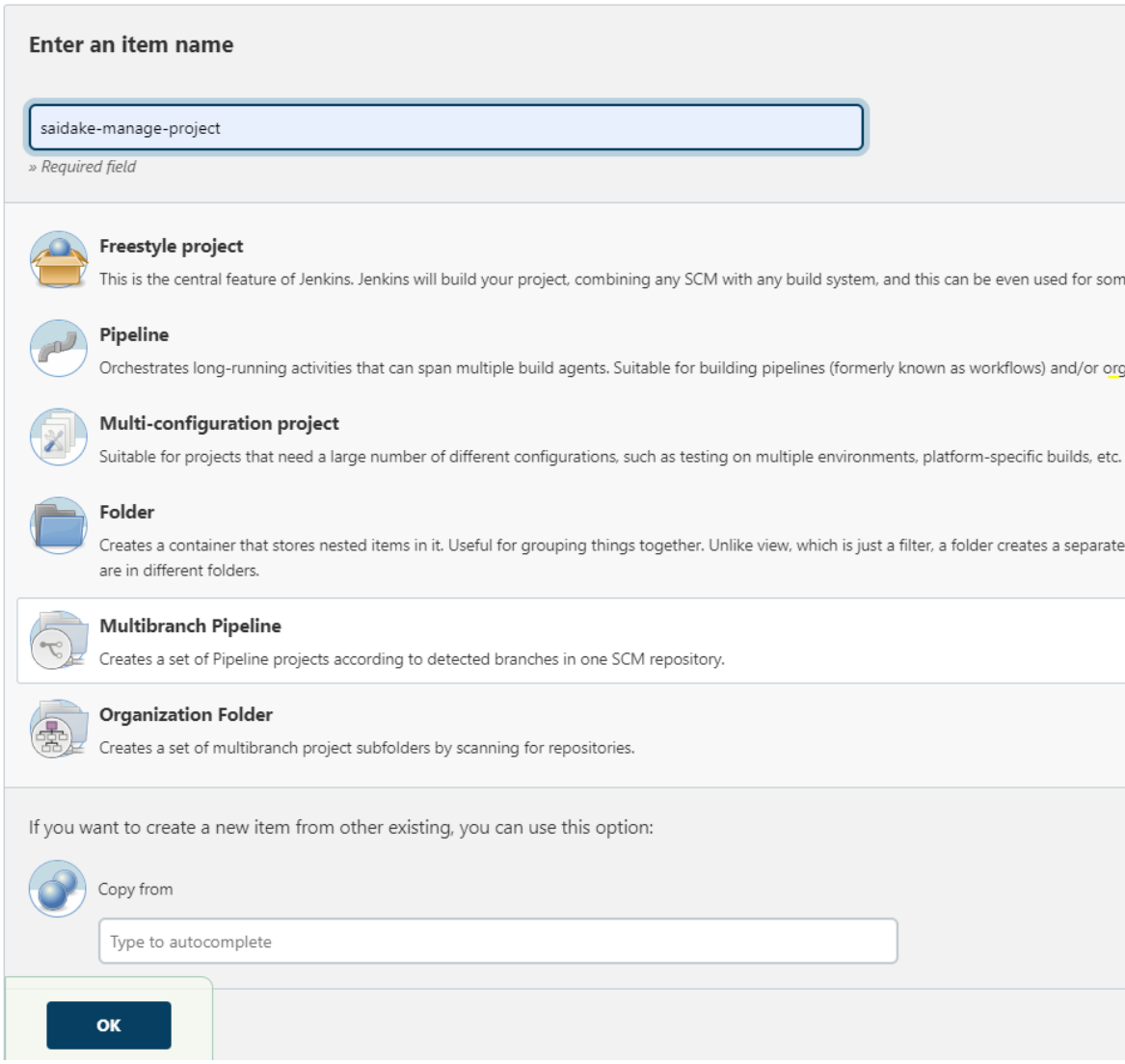
项目类型

Multibranch Pipeline 基于 Git 分支自动创建 Jenkins Pipeline。可以在 SCM 中创建时自动发现新的 Git 分支，并自动为该分支创建管道。

当管道构建开始时，使用当前分支的 Jenkinsfile 进行构建阶段。（SCM 可以是 Github，Bitbucket 或 Gitlab 存储库）

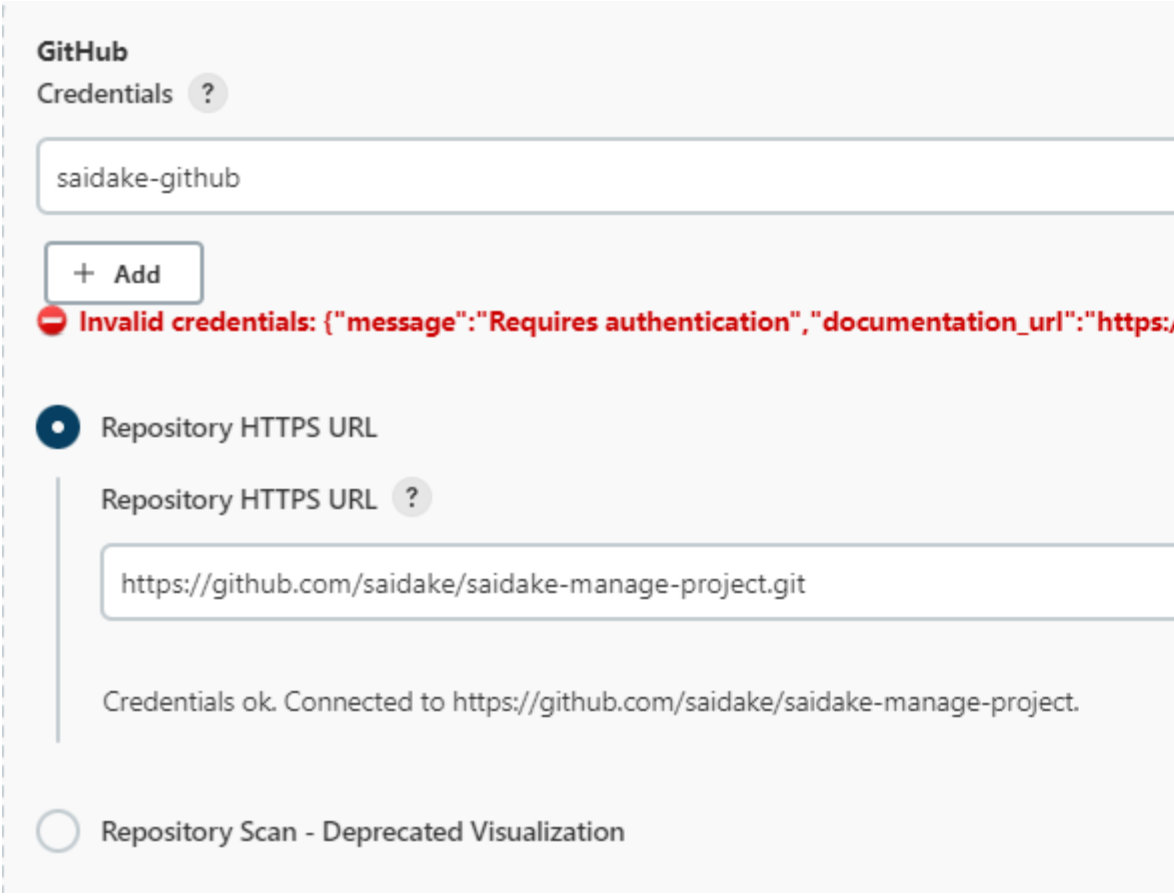
多分支管道支持基于 PR 的分支发现。这意味着，如果有人从分支提出 PR（拉动请求），则会在管道中自动发现分支。如果启用了此配置，则仅在提 PR 时才会触发构建。

创建：

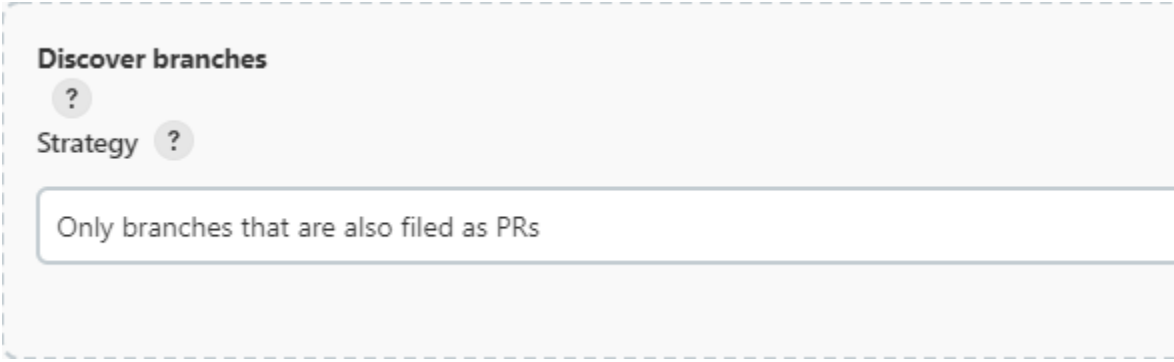


Branch Sources 配置：

git 仓库

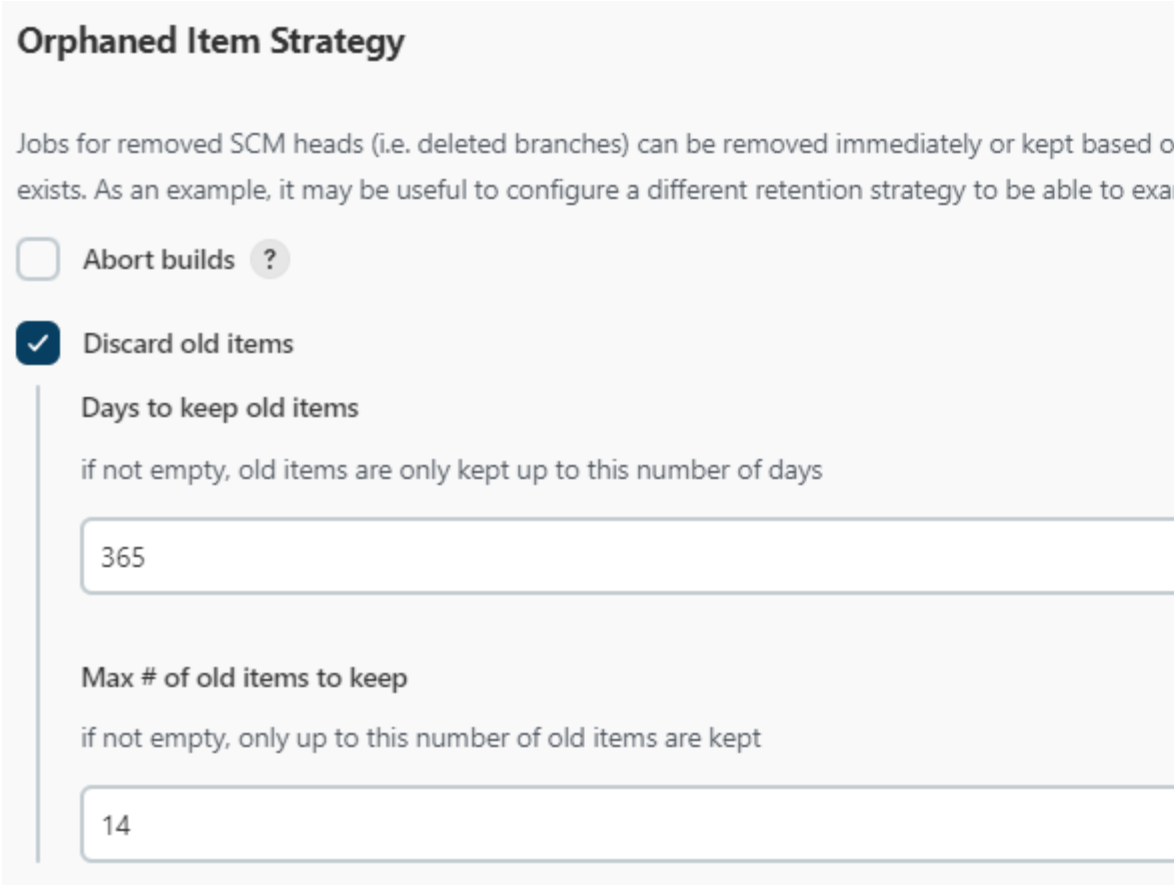


如何发现分支（发现存储库中的所有分支，也可以仅选择具有"拉取请求"的分支，管道还可以从分叉的仓库中发现具有 PR 的分支）



手动添加一个规则，通过正则表达式或通配符方法从存储库中发现分支

保留策略



介绍

开源的 java 语言开发持续集成工具，支持持续集成，持续部署，一个持续集成服务器。

消息通知及测试报告：集成 RSS/E-mail 通过 RSS 发布构建结构或当构建完成时通过 e-mail 通知，生成 JUnit/TestNG 测试报告

分布式构建：支持 jenkins 能够让多台计算机一起构建/测试（注意工具和环境变量都需要配置）

文件识别：jenkins 能够跟踪哪次构建生成哪些 jar，哪次构建使用哪个版本的 jar 等

构建项目类型：

自由风格软件项目（FreeStyle Project）

Maven 项目（Maven Project）

流水线项目（Pipeline Project）

软件开发生命周期 SDLC：需求分析，设计系统架构部署架构，实现代码，测试，优化维护

软件开发瀑布模型：流水线式开发，不便于回退流程

敏捷开发：核心是迭代开发和增量开发

持续集成：频繁地将代码集成到主干（让产品可以快速迭代，同时还能保持高质量）

降低风险，由于持续集成不断去构建，编译和测试，可以很早期发现问题

对系统健康持续检查，减少发布风险带来的问题

Pipline

Pipline 介绍：是由 Groovy 语言实现，

支持两种语法：Declarative 声明式和 Scripted Pipline 脚本式语法

可以直接在 Jenkins 的 webUI 界面中输入脚本，也可以通过创建一个 Jenkinsfile 脚本文件放入项目源码库中（一般都推荐在 Jenkins 中直接从源代码控制 SCM 中直接载入 Jenkinsfile Pipeline 这种方法）

jenkins 首页 Configure System 内定义的环境变量【BUILD_ID，BUILD_NUMBER，JENKINS_URL，JOB_NAME】

Jenkinsfile (Scripted Pipline syntax)

pipeline {

```
agent any                              代理环境
                                           any 运行在任一可用节点
                                           none 当 pipeline 全局指定 agent 为 none，则根据每个 stage 中定义的 agent 运行（stage 必须指定）
                                           label 在指定的标签/分组的节点运行
                                           node 支持自定义流水线的工作目录
agent { label "xxxlabelName" }

agent {              自定义节点，加上 node 有一些高级的特性，比如自定义工作目录
    node {
        label "labelName",
        customWorkspace "/opt/agent/workspace"
    }

paramenters {                          参数
    booleanParam(name: 'DEBUG_BUILD', defaultValue: true, description: '')                                  布尔参数，用户可以在 Jenkins UI 上选择是还是否
    choice( name:'ENV_TYPE',  choices:['test', 'dev', 'product'],  description:'test means test env,...')      多选参数，支持用户从多个选择项中，选择一个值用来表示这个变量的值
    string( name: 'DEPLOY_ENV',  defaultValue: 'staging', description: '')                                   字符串参数，定义一个字符串参数（用户可以在 Jenkins UI 上输入字符串）
    text(name: 'Welcome_text', defaultValue: 'One\nTwo\nThree\n', description: '')                           多行字符串参数，支持写很多行的字符串
    password( name:'CredsToUse',  description:'Apassword to build with',  defaultValue:'')                 在 Jenkins 参数化构建 UI 提供一个暗文密码输入框
    name: 'FILE', description: 'Some file to upload')                                                       在 Jenkins 参数化构建 UI 上提供一个文件路径的输入框，Jenkins 会自动去你提供的网络路径去查找并下载。一般伴随着还有你需要在 Pipleline 代码中写解析文件
}
options {                              运行时选项
    timeout(time: 50, unit: 'MINUTES')        设定流水线的超时时间(可用于阶段级别)
    timestamps()                  设置日志时间输出(可用于阶段级别)
    retry(3)                      设定流水线的重试次数(可用于阶段级别)
    skipDefaultCheckout()   跳过默认的代码检出
    disableConcurrentBuilds()    禁止并行构建，因为用的 worksapce 还是同一个
    buildDiscarder(logRotator(numToKeepStr: '1'))       设置保存最近的记录
}
environment {                          定义环境变量
    SYSTEM_PARA ='abc${paramers.PerformMavenRelease}'
    ONE_ACCESS_KEY = credentials(saidake-credit)      获取 jenkins 预先定义的凭证
    REGEX='(master|release/.+)'
}
stages {
    stage('Build maven') {         多步执行
        when { anyOf {branch 'master'; environment name:'ENV_PARAM', value:'env-param' }  }       满足任何一个条件成立
        when { allOf {branch 'master'; environment name:'ENV_PARAM', value:'env-param' }}          条件都成立执行
        when { not {branch 'master'}}                                                              条件不成立执行
```

```
            when { branch 'master'}                                          是 master 分支时执行
            when { expression { return env. LS_GIT_BRANCH==~"release.*" }}     表达式求值为 true 时执行
            steps {                  stages 里必须有 steps
                script {            执行 goovy 脚本
                    int x = 5
                    long y = 100L
                    float a = 10.56f
                    double b = 10.5e40
                    BigInteger bi = 30g
                    BigDecimal bd = 3.5g
                    def username ='JENKINS'            goovy 定义变量
                    println("--- test content ---")

                    def browsers = ['chrome','firefox']
                    for (int i = 0;i < browers.size();++i){
                        echo "Testing the ${browsers[i]} browser"
                    }
                    echo "The job name is ${env.JOB_NAME}"       //内置的环境变量
                    env.JOB_NAMEA="mytest-pipeline"               //自定义的全局变量,也就是整个流水线可以去使用
                }

                echo 'xxx ${env.BUILD_ID} ${SYSTEM_PARA} '        执行 linux 服务器命令【 ${env.xxx} 使用环境变量或者自定义
的变量：JOB_NAME 项目名称，JENKINS_URL 完整 jenkins 网址 】
                sh 'cd /loopo'            执行命令
            }
        }
        stage('Sonar 扫描 ') {
            steps {
                script {
                    scannerHome = tool 'sonarScannerTool'        //使用 jenkins 配置的工具
                    mavenHome = tool 'mavenTool'
                }
                withSonarQubeEnv('sonarqubeServer') {              //使用 jenkins 的全局配置
                    sh "cd loopo; ${mavenHome}/bin/mvn compile;cd .."
                    sh "${scannerHome}/bin/sonar-scanner;"
                }
            }
        }
    }
    post {
        always {                        执行结束总是执行
            emailext (                  发送邮件
                subject: "构建通知：  ${JOB_NAME} - 第${BUILD_NUMBER}次构建记录",
                body: '${FILE,path="email.html"}',
                to: "saidake@qq.com"
            )
        }
```

```
        faiure {                执行失败
            echo 'xxx '
        }
        success {                执行成功
            echo 'xxx '
        }
        cleanup {            其他 post 条件执行结束后
            echo 'xxx '
         }
    }
    triggers{                    流水线项目应该怎样重新触发
        pollSCM 'H/2 * * * *'
    }
}

def   someGlobalFunction() {
     // common code
}
```

Declarative 声明式语法：

```
        def git_auth="xx"              //定义变量
        node {
            def mvnHome                 //定义变量
            stage('pull code'){
                def scannerHome = tool 'sonarqube-scanner'          //引入 jenkins 全局工具
                withSonarQubeEnv('sonarqubeServer'){                //引入 jenkins 全局环境
                    sh "cd ${project_name};${scannerHome}/bin/sonar-scanner"
                }
            }
            stage('Build maven'){
            }
            stage('Build maven'){
            }
        }
```

项目创建

Pipeline 流水线部署（Pipeline）

1. 创建流水线项目，添加 git 地址，项目的 Jenkinsfile 或直接 jenkins 网页配置执行代码
2. build 测试能否拉到代码，打印 pipeline 代码【完毕可以准备 sonar 环境】
2. 参照官方文档生成 Pipeline Snippet Git 脚本，运行 echo 测试，拉取代码（https://www.jenkins.io/doc/book/pipeline/）

高级项目选项（替代了拉取代码等操作）【try sample Pipeline】

Pipline script from SCM　　通过项目脚本文件 build

流水线脚本生成器 Jenkinsfile

maven 项目部署 Deploy to container 插件，点击项目，配置，构建：（Maven Integration）

1. 构建触发器：增加本机构建步骤 shell　　mvn clean package
2. 选择 Pre Steps，Build，Root POM==pom.xml，Goals and options==clean package
3. 构建后操作 Deploy war/ear to containr　(WAR/EAR files: target/*.warFF09

自由项目部署 Deploy to container 插件，点击项目，配置，构建：　(Freestyle Project)

1. 项目配置：构建后操作选择 Deploy war/ear to container，添加 Credentials 和 Tomcat URL

项目构建

Jenkins 内置构建触发器：

触发远程构建（其他项目可以通过加密字符串　访问远程地址构建）

其他工程构建后出发

定时构建　　H/30****　　每个小时的 H 分，每 30 分钟构建一次 10:02 -->10:32（H 代表形参）

H H/2***　　每两小时构建一次

0 8, 12, 22***　每天的 8 点，12 点，22 点，一天构建 3 次

H 12***　　每天中午 12 点定时构建一次

H 18***　　每天下午 18 点定时构建一次

H(0-29)/10****　每个小时的前半个小时内的每 10 分钟

H H(9-16)/2**1-5　每两小时一次，每个工作日上午 9 点到下午 5 点

轮询 SCM　　定时扫描代码仓库代码变更，定时扫描整个项目代码，不建议使用

GIT hook 自动触发构建（Gitlab Hook 和 Gitlab 插件，GItlab 代码变更时会发送构建请求）

1.添加构建触发器

2. git lab 配置钩子请求此网址

3. 配置认证：Config System --> Enable authenication for /project end-point 不能勾选

参数化构建：　1. 构建时添加参数【choice parameter：回车添加多个选项】

2. Jenkinsfile 访问参数${branch}

非流水线项目审查：

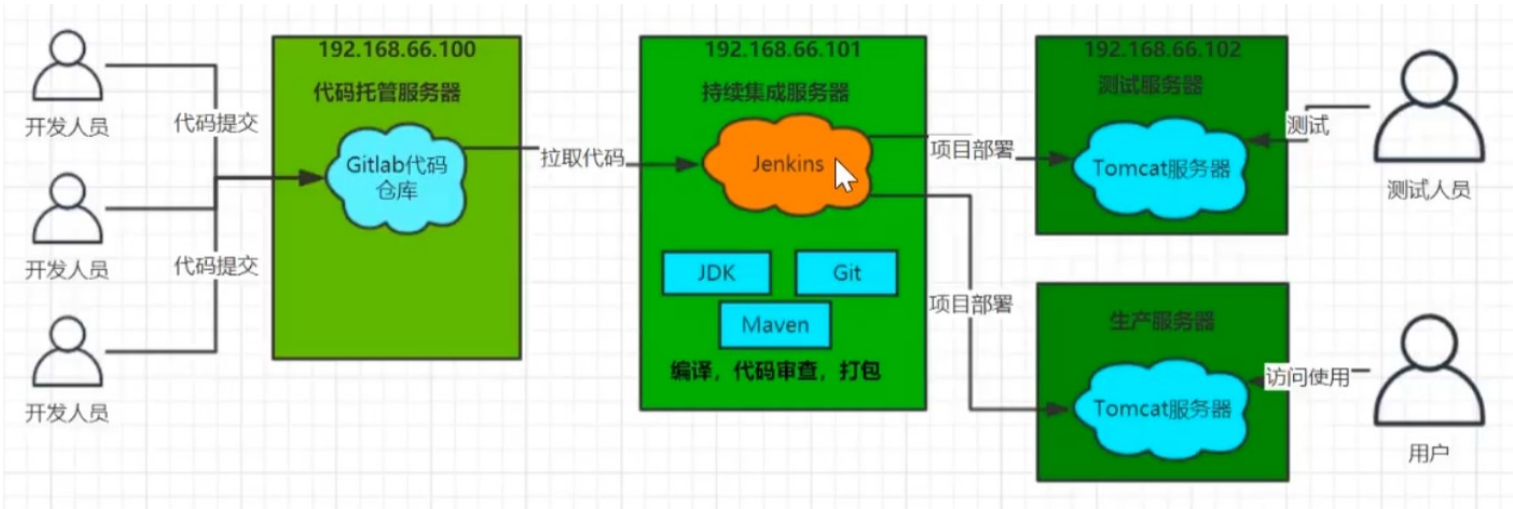
1. 项目配置：添加构建步骤　sonarQube Scanner

流水线项目审查：

1. 添加 sonarqube 配置文件

2. Jekninsfile 添加代码审查，tool + Jenkins 的 sonarqube 插件名称　withsonarqubeenv　jenkins 的连接 sonar 服务名称配置

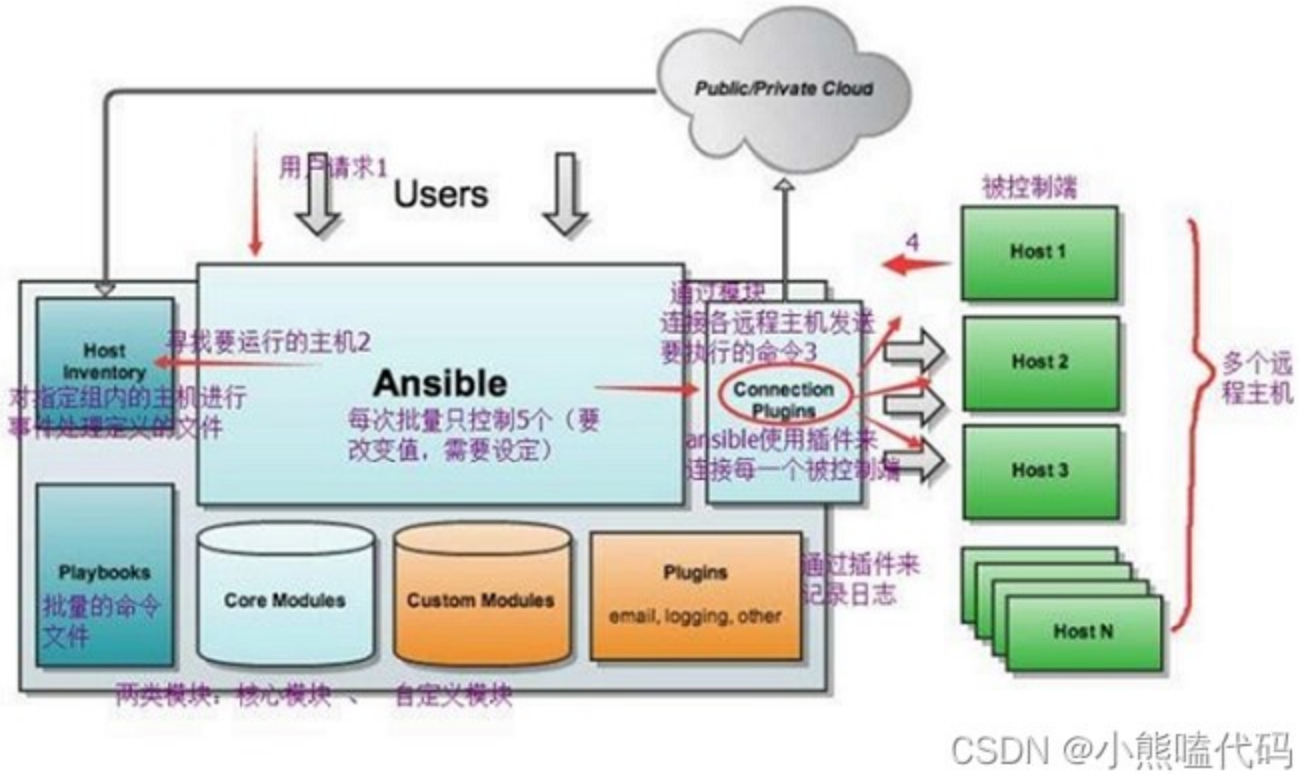
持续集成流程：



openshift

Ansible

Ansible：新出现的自动化运维工具，基于 Python 开发，集合了众多运维工具（puppet、chef、func、fabric）的优点，实现了批量系统配置、批量程序部署、批量运行命令等功能。



执行模式

adhoc (点对点模式)

使用单个模块，支持批量执行单条命令。

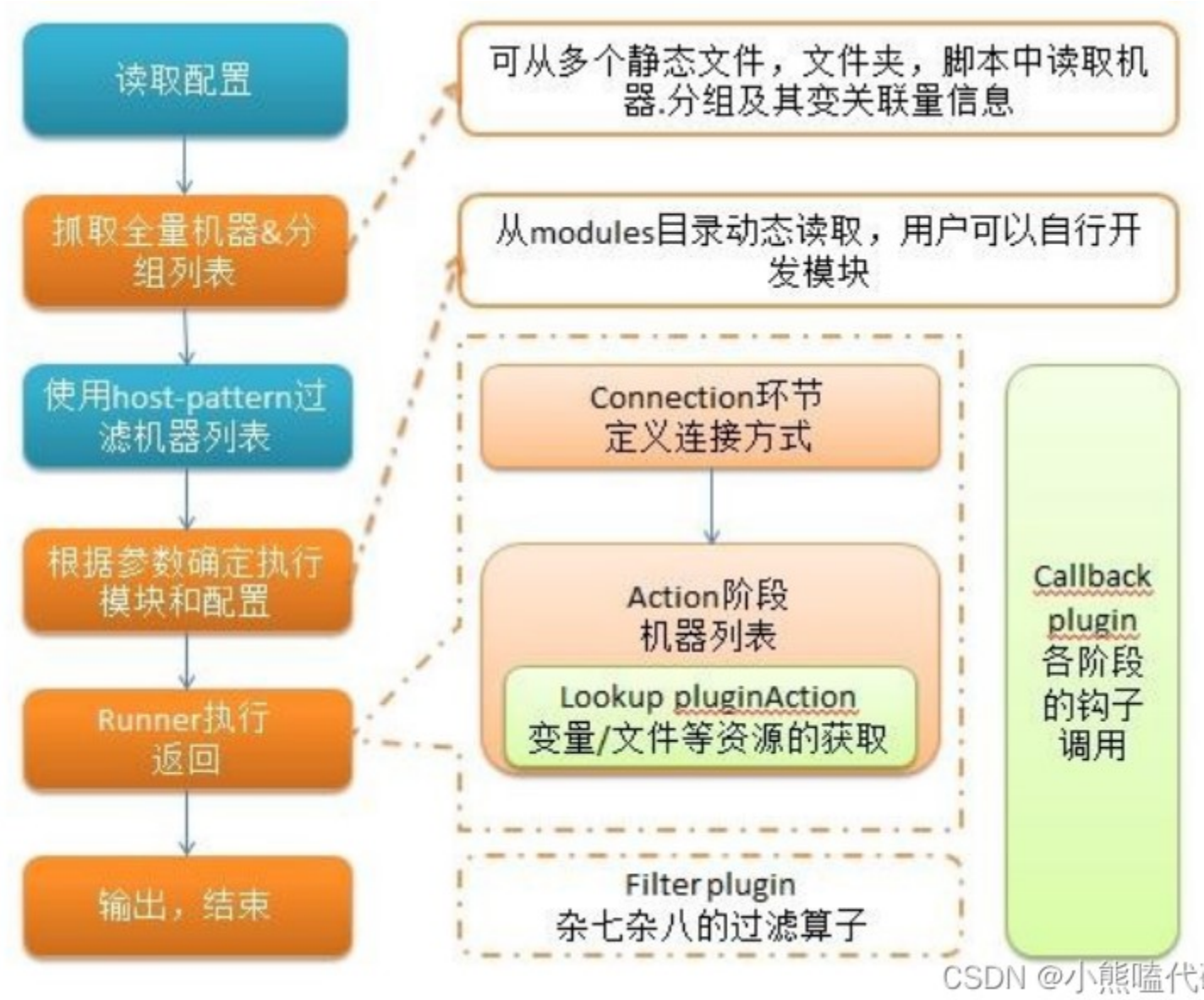
ad-hoc 命令是一种可以快速输入的命令，而且不需要保存起来的命令。就相当于 bash 中的一句话 shell。

playbook(剧本模式)

是 Ansible 主要管理方式，也是 Ansible 功能强大的关键所在。

playbook 通过多个 task 集合完成一类功能，如 Web 服务的安装部署、数据库服务器的批量备份等。

可以简单地把 playbook 理解为通过组合多条 ad-hoc 操作的配置文件。



命令

ansible-playbook config_hosts.yml　　执行剧本　　【-i xxinventory 仓库　　-e xxvars　　额外传递变量　　-u xxuser　指定运行用户身份

【 -C　检测运行，不是真正运行　　-t　　指定执行 tags 段 】

概念

OpenShift：每台机器都使用一个配置启动，该配置引用托管在它加入的集群中的资源。这允许集群在应用更新时进行自我管理。

这种方法的一个缺点是，如果没有外部帮助，新集群无法启动——待创建集群中的每台机器都在等待待创建集群。

安装程序生成的主要资产是 bootstrap、master 和 worker 机器的 Ignition 配置。鉴于这三个配置（以及正确配置的基础设施），可以启动 OpenShift 集群。引导集群的过程如下所示：

bootstrap 机器启动并开始托管 master 机器启动所需的远程资源。

master 机器从 bootstrap 机器获取远程资源并完成引导。

master 机器使用 bootstrap 节点形成 etcd 集群。

引导节点使用新创建的 etcd 集群启动一个临时的 Kubernetes 控制平面。

临时控制平面将生产控制平面调度到主机。

引导节点通过临时控制平面注入特定于 OpenShift 的组件。

临时控制平面关闭，只留下生产控制平面。

安装程序会拆除引导程序节点。

安装程序可以创建以下目标：

install-config　　安装配置包含安装过程的主要参数。此配置为用户提供了比交互式提示更多的选项，并预先填充了默认值。

manifests　　此目标输出将安装在集群上的所有 Kubernetes 清单。

ignition-configs　　这些是 bootstrap、master 和 worker 机器的三个 Ignition Configs。

cluster　　该目标提供集群及其关联的基础设施。

安装程序可以销毁以下目标：

cluster　　这会破坏创建的集群及其关联的基础设施。

bootstrap　　这会破坏引导基础结构。

minishfit　查看版本：https://github.com/minishift/minishift/releases

wget https://github.com/minishift/minishift/releases/download/v1.34.3/minishift-1.34.3-linux-amd64.tgz

linux 安装

安装指南：https://f5-agility-labs-containers.readthedocs.io/en/develop/class4/module1/lab2.html

官方文档推荐开启 SELINUX，否则会导致安装失败

/etc/selinux/config >>

SELINUX=enforcing

SELINUXTYPE=targeted

所有主机下安装 OpenShift 依赖的软件包

yum -y install wget git net-tools bind-utils iptables-services bridge-utils bash-completion kexec-tools sos psacct bash-completion.noarch python-passlib NetworkManager vim libselinux-python

在 openshift 源文件根目录执行检查 ansible-playbook　/usr/local/openshift-ansible-3.11.934-1/playbooks/prerequisites.yml

配置

ansible.cfg >>　　定义 hosts 仓库

```
[defaults]
log_path = ~/openshift-ansible.log            # 日志路径
forks = 20
host_key_checking = False
retry_files_enabled = False
retry_files_save_path = ~/ansible-installer-retries
nocows = True
remote_user = root
roles_path = roles/
gathering = smart
fact_caching = jsonfile
fact_caching_connection = $HOME/ansible/facts
fact_caching_timeout = 600
callback_enabled = profile_tasks          #callback_whitelist = profile_tasks
inventory_ignore_extensions = secrets.py, .pyc, .cfg, .crt, .ini
timeout = 30                                  # work around privilege escalation timeouts in ansible:

inventory = inventory/hosts.example        # 定义 hosts 仓库

[inventory]
unparsed_is_failed=true                       # inventory 必须解析通过
```

```
 [ssh_connection]
pipelining = True
ssh_args = -o ControlMaster=auto -o ControlPersist=600s
timeout = 10
control_path = %(directory)s/%%h-%%r          # 缩短 ControlPath
```

inventory/hosts >>

这是一个 OpenShift Ansible 主机清单的示例，它提供生产使用的最低推荐配置。包含内容：

- 3 个主机
- 两个 infra 节点
- 两个计算节点
- 一个要加载的 haproxy 负载平衡器（平衡到 API 服务器的流量。对于真正的生产环境，应该使用一个自身高度可用的外部负载平衡解决方案。）

```
[masters]
ose3-master[1:3].test.example.com

[etcd]
ose3-master[1:3].test.example.com

[nodes]
ose3-master[1:3].test.example.com openshift_node_group_name="node-config-master"          # openshift_node_group_name 必须为每个节点提供，
ose3-infra[1:2].test.example.com openshift_node_group_name="node-config-infra"
ose3-node[1:2].test.example.com openshift_node_group_name="node-config-compute"

[nfs]
ose3-master1.test.example.com

[lb]
ose3-lb.test.example.com

 [OSEv3:children]       #创建一个 OSEv3 组，包含主节点和节点组
masters
nodes
etcd
lb
nfs

[OSEv3:vars]          #公共配置项
ansible_user=root      # ssh 用户,这个用户允许基于权限的 ssh 不需要密码,如果使用基于权限的 sshkey,则这个 key 应该被 ssh agent 管理

#ansible_become=yes   #如果可解释用户不是 root，可解释必须是 true，并且这个用户必须为了无密码而配置
#openshift_deployment_type=openshift-enterprise
#openshift_release="3.11"
```

openshift_master_default_subdomain=apps.test.example.com
openshift_master_cluster_hostname=ose3-lb.test.example.com
debug_level=2
#openshift_image_tag=v3.11.0
#openshift_pkg_version=-3.11.0
#system_images_registry="docker.io"
#openshift_install_examples=true
#openshift_master_logout_url=http://example.com
#openshift_master_oauth_templates={'login': '/path/to/login-template.html'}
openshift_master_oauth_template is deprecated. Use openshift_master_oauth_templates instead.
#openshift_master_oauth_template=/path/to/login-template.html

命令

oc login --token-sha25638id8C4ZTAOjSqHKdsnO0zXYEDQN-voN7S59iRStkFxUe --server-https://api.namicgswd25u.ecs.dyn.nsroot .net:6l43 使用 token 登录项目

oc status 查看登录在哪个 project 下，并查看当前项目状态

oc project xxproject 切换到 project

oc get all 查看当前 project 下的所有资源

oc get nodes 获取集群所有节点

STATUS 类型：

Ready	节点通过返回 StatusOK 来通过 master 节点执行的健康检查
NotReady	节点未通过主节点执行的健康检查
SchedulingDisabled	无法安排 Pod 放置在节点上

NAME	STATUS	ROLES	AGE	VERSION
master.lab.example.com	Ready	master	17d	v1.9.1+a0ce1bc657
node1.lab.example.com	Ready	compute	17d	v1.9.1+a0ce1bc657
node2.lab.example.com	Ready	compute	17d	v1.9.1+a0ce1bc657

oc get node xxnode1.lab.example.com 查看单个节点信息

NAME	STATUS	ROLES	AGE	VERSION
node1.lab.example.com	Ready	compute	17d	v1.9.1+a0ce1bc657

oc get pods -n xxnamespace 查看当前 project 下的所有 pod【-o wide】

oc get limitrange -n xxnamespace 获取对应 namespace 的 limitrange 配置文件

oc get pod 查看该 project 下的 pod

oc get service 查看该 project 下的 service

oc get endpoints 查看该 project 下的端点，支持服务的服务器称为其端点，并且由具有与服务 相同名称的类型端点 的对象指定。当服务由 pods 支持时，服务规范中通常由标签选择器指定这些 pods，而 OpenShift 容器平台会自动创建指向这些 pods 的端点对象。

oc get scc 查看 scc

oc describe node xxnode1.lab.example.com 查看对应节点详细信息，可以看到运行在该节点下的 pod

Name: node1.example.com # 节点的名称

Roles: master # 节点的角色，要么 master

要么 compute
Labels: beta.kubernetes.io/arch=amd64 # 应用于节点的标签
Annotations: volumes.kubernetes.io/controller-managed-attach-detach=true
应用于节点的注释
Taints: <none> # 应用于节点的污点
Conditions: # 节点条件
Addresses: # 节点的 IP 地址和主机名
Capacity: # pod 资源和可分配资源
System Info: # 有关节点主机的信息
....
Non-terminated Pods: (6 in total) # 节点上的 pod

oc describe pod xxpod -n xxnamespace 查看 pod 详细信息
oc describe limitrange limitrange.config -n xxnamespace 查看配置文件详情

oc logs xxpod -c xxcontainer_name -f 查看 pod log【-f 输出是否遵循要写到日志中的内容】
oc logs deployment/ruby 指定资源类型和名称

oc rollout history DeploymentConfig/xxdc 查看 dc 历史版本
oc rollout history DeploymentConfig/xxdc --revision=5 回滚到 5 版本
oc rollout latest "deploy-config-example"

oc edit limitrange limitrange.config -n xxnamespace 修改 limitrange 配置
oc adm policy add-scc-to-user anyuid -z default 为该 project 下 default 账户开启 anyuid，可以使用 root 权限，一般是安装/运行某些软件时需要
oc adm policy remove-scc-from-user anyuid -z default 删除 default 的 anyuid 权限
oc delete pod xxpod -n xxnamespace 重启 pod
oc delete node xxnode 删除 node，删除节点后将只为处于节点状态是 Ready 的节点上调度 Pod
oc api-resources 查看 server 支持的 api 资源

oc scale dc xxpod --replicas=3 -n namespace 设置副本数为 3
oc autoscale dc xxdc --min=2 --max=10 --cpu-percent=80 设置自动伸缩最小 2，最大 10
oc describe scc anyuid 查看 anyuid 详细信息,user 即包含已经开启 anyuid 的 project
oc describe clusterrole.rbac 查看集群管理员角色及权限
oc describe clusterrolebinding.rbac 查看用户组及绑定的角色
oc adm policy add-cluster-role-to-user cluster-admin username 添加 username 为 cluster-admin
oc get routes --all-namespaces 查看所有 namespace 的 route
docker ps -a|grep xxpod 查看 pod 对应 containerID
docker exec -it containerID /bin/sh 登录到 container
oc new-project my-project 创建 project
oc new-app https://github.com/sclorg/cakephp-ex 创建 APP
oc adm must-gather //收集当前集群的状态信息
oc adm top pods //查看 pod 资源状态
oc adm top node //查看节点资源状态
oc adm top images //查看 images 使用情况

oc adm cordon node1 //标记 node1 为 SchedulingDisabled
oc adm manage-node <node1> --schedulable = false //标记 node1 为 unschedulable
oc adm manage-node node1 --schedulable //标记 node1 为 schedulable
oc adm drain node1 //将 node1 进入维护模式
oc delete node //删除 node
oc adm node-logs --role master -u NetworkManager.service //获取节点网络日志
oc get csr //查询 CSR(certificate signing requests)
oc adm certificate approve csr-name //approve CSR
oc adm certificate deny csr_name //拒掉 csr
oc get csr|xargs oc adm certificate approve csr //approve 所有 csr
echo 'openshift_master_bootstrap_auto_approve=true' >> /etc/ansible/hosts //设置自动 approve csr
oc get project projectname //get project
oc describe project projectname //查看 project 信息
oc get pod xxpod -o yaml //查看 xxpod yaml 文件
oc get nodes --show-labels //查看节点 label
oc label nodes xxnode label-key=label-value //给 node 添加标签，pod 也要有对应 label（在第二层 spec 下添加 nodeSelector），pod 就会运行在指定的 node 上
oc label nodes xxnode key=new-value --overwrite //更新指定 key 值的 label
oc label nodes xxnode key- //删除指定 key 值的 label
oadm manage-node xxnode --list-pods //查看运行在某 node 上的所有 pod
oadm manage-node xxnode --schedulable=false //mark node as unschedulable
oc get svc //查看 service
oc patch dc xxdc -p '{"spec":{"template":{"spec":{"containers":[{"name":"xxdc","securityContext":{"runAsUser": 1000160000}}],"securityContext":{"runAsUser": 1000160000,"fsGroup": 1000160000}}}}}' //修改 xxdc yaml 文件
oc login --token=iz56jscHZp9mSN3kHzjayaEnNo0DMI_nRlaiJyFmN74 --server=https://console.qa.c.sm.net:8443 //使用命令行登录 openshift，token 是用你自己的账户在登录网址时生成的 token
oc rsh -t //查看 token
oc rsh xxpod //使用命令行登录 pod
oc whoami --show-server //查看当前登录的服务地址
oc whoami //查看当前登录账户
oc get dc -n namespace //查看 deployment config
oc get deploy -n namespace //查看 deploy
oc edit deploy/deployname -o yaml -n namespace //编辑 deploy yaml 文件
oc get cronjob //查看 cronjob
oc edit cronjob/cronjob-name -n xxnamespace //编辑 cronjob
oc describe cm/configmap-name -n xxnamespace //查看 cronjob
oc get configmap -n xxnamespace //查询 configmap
oc get cm -n xxnamespace //查询 configmap
cat /etc/origin/master/master-config.yaml|grep cidr //查看集群 pod 网段规划
oc edit vm appmngr54321-poc-msltoibh -n appmngr54321-poc -o yaml //编辑 VM yaml 文件
oc serviceaccounts get-token sa-name //获取 sa-name 的 token
oc scale deployment deployment-name --replicas 5 //扩展 pod 副本数量
oc config view //查看 config
TOKEN=$(oc get secret $(oc get serviceaccount default -o jsonpath='{.secrets[0].name}') -o jsonpath='{.data.token}' | base64 --decode) //get token
APISERVER=$(oc config view --minify -o jsonpath='{.clusters[0].cluster.server}') //get apiserver

```
curl $APISERVER/api --header "Authorization: Bearer $TOKEN" --insecure   //curl API
oc api-versions   //get api versions
oc api-resources //get api-resources
oc get hpa   --all-namespaces//查询 HPA
oc describe hpa/hpaname -n namespace   //查看 HPA，可以看到 Metrics，Events
oc create serviceaccount caller //创建 sa
oc adm policy add-cluster-role-to-user cluster-admin -z caller //赋予 cluster-admin 权限
oc serviceaccounts get-token caller //get sa token
echo $KUBECONFIG //获取 kubeconfig 文件位置
oc get cm --all-namespaces -l app=deviation //根据 labelselector 筛选 cm
oc describe PodMetrics podname   //查询 pod CPU/mem usage，openshift4.X 适用
oc api-resources -o wide    //查看 shortnames、apiGroup、verbs
oc delete pod --selector logging-infra=fluentd //按 selector 删除
oc get pods -n logger -w   //watch  某个 pod 状态
oc whoami   //查看当前登录账户
oc explain pv.spec   //查看资源对象的定义
oc get MutatingWebhookConfiguration    //查看 MutatingWebhook
oc get ValidatingWebhookConfiguration   //查看 ValidatingWebhook
oc annotate validatingwebhookconfigurations <validating_webhook_name> service.beta.openshift.io/inject-cabundle=true
//给 validatingwebhook 注入 CA
oc annotate mutatingwebhookconfigurations <mutating_webhook_name> service.beta.openshift.io/inject-cabundle=true
//给 mutatingwebhook 注入 CA

oc get secrets/signing-key -n openshift-service-ca \
       -o template='{{index .data "tls.crt"}}' \
       | base64 -d \
       | openssl x509 -noout -enddate
//查看当前服务 CA 证书的到期日期
oc delete secret/signing-key -n openshift-service-ca   //手动更新该服务证书

for I in $(oc get ns -o jsonpath='{range .items[*]} {.metadata.name}{"\n"} {end}'); \
         do oc delete pods --all -n $I; \
         sleep 1; \
         done
//将新证书应用到所有服务，请重启集群中的所有 pod

oc annotate crd <crd_name> \


service.beta.openshift.io/inject-cabundle=true    //给 CRD 注入 CA
oc annotate apiservice <api_service_name> service.beta.openshift.io/inject-cabundle=true    //给 apiservice 注入 CA
oc annotate configmap <config_map_name> service.beta.openshift.io/inject-cabundle=true    //给 configmap   注入 CA
oc annotate service <service_name> service.beta.openshift.io/serving-cert-secret-name=<secret_name>    //给 service 添加
服务证书
oc get pv --selector=='path=testforocp' //根据 label 查询 pv

# oc get --raw "/apis/metrics.k8s.io/v1beta1" | jq .
```

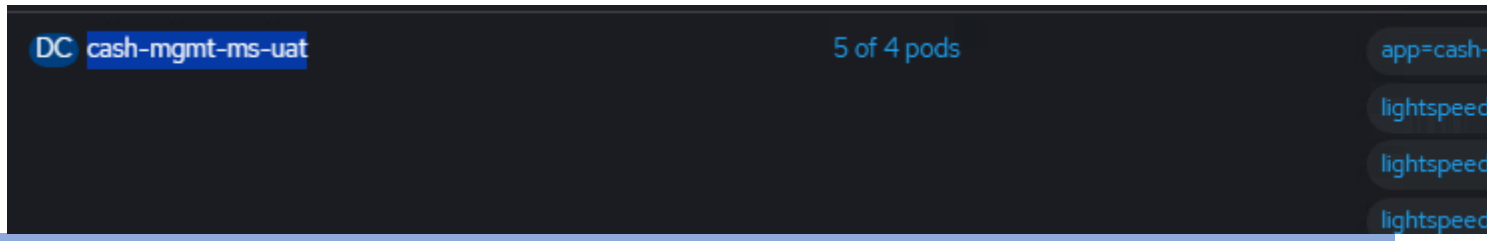
```
{
  "kind": "APIResourceList",
  "apiVersion": "v1",
  "groupVersion": "metrics.k8s.io/v1beta1",
  "resources": [
    {
      "name": "nodes",
      "singularName": "",
      "namespaced": false,
      "kind": "NodeMetrics",
      "verbs": [
        "get",
        "list"
      ]
    },
    {
      "name": "pods",
      "singularName": "",
      "namespaced": true,
      "kind": "PodMetrics",
      "verbs": [
        "get",
        "list"
      ]
    }
  ]
}
```

```
 oc get cm --all-namespaces -o=jsonpath='{.items[?(@..metadata.annotations.cpuUsage=="0")].metadata.name}'   //根据自定义 annotations 查询，返回 cm name
ns=$(oc get cm --selector='app=dev' --all-namespaces|awk '{print $1}'|grep -v NAMESPACE)
for i in $ns;
> do
> oc get cm dev -oyaml -n $i >> /tmp/test.txt
> done;

# oc project openshift-etcd
# oc rsh etcd-master-0.ocp4.example.com     //进入 etcd
# etcdctl get /kubernetes.io/ --prefix --keys-only   //查询资源
# etcdctl get /kubernetes.io/horizontalpodautoscalers/yunxiao5432/example12345 --prefix   //查询 HPA
# oc get --raw /apis/autoscaling/v2beta1/namespaces/<namespace>/horizontalpodautoscalers/<resource_name> | jq .
        查询 HPA   注：使用哪个版本(v2beta1)查询出来的就是哪个版本，查询时自动转换
# oc rollout undo deploy webapp   --to-revision=2 //回退至指定版本
# oc rollout status deploy webapp   //刷新状态
# oc rollout history deploy webapp //list 历史版本
# oc rollout pause deploy webapp //暂停 rollout
# oc rollout resume deploy webapp //恢复 rollout
```

Workloads

DeploymentConfigs

Start rollout　　Restart application



sonarQube

sonarqube 安装

wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.2.0.32929.zip

unzip sonarqube-8.2.0.32929.zip -d /opt

vi /opt/sonarqube-8.2.0.32929/conf/sonar.properties　　修改配置

useradd sonarqube

chown -R sonarqube.sonarqube /opt/sonarqube-8.2.0.32929/

sudo -u sonarqube /opt/sonarqube-8.2.0.32929/bin/linux-x86-64/sonar.sh console　　测试执行

sonarqube 配置：vi /opt/sonarqube-8.2.0.32929/conf/sonar.properties

sonar.jdbc.username=sonarqube

sonar.jdbc.password=abc-123

sonar.jdbc.url=jdbc:postgresql://172.16.254.129/sonarqube

sonar.web.host=0.0.0.0

sonar.web.port=9000

sonarqube 配置：vi /opt/sonarqube-8.2.0.32929/conf/wrapper.conf

wrapper.java.command=/usr/java/jdk1.8.0_72/bin/java　　配置 jdk 启动环境

sonar scanner 安装

wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-3.3.0.1492-linux.zip

unzip sonar-scanner-cli-3.3.0.1492-linux.zip

mv sonar-scanner-3.3.0.1492-linux/ /usr/local/

vi /usr/local/sonar-scanner-3.3.0.1492-linux/conf/sonar-scanner.properties

sonar.host.url=http://so.stgame.cn:48080　　// sonar 网址

sonar.sourceEncoding=UTF-8　　// 默认编码

sonar.jdbc.username=admin　　// 数据库用户名

sonar.jdbc.password=sonar　　// 数据库密码

sonar.jdbc.url=jdbc:mysql://10.9.40.121:3307/sonar?useUnicode=true&characterEncoding=utf8　　// 数据库连接地址

sonar.login=admin　　// Sonar 用户名

sonar.password=qew3dqr4w　　// Sonar 密码　　【配置完后在项目内测试】

4. sonar scanner 环境变量：　export SONAR_SCANNER_HOME=/usr/local/sonar-scanner-3.3.0.1492-linux

export PATH=$SONAR_SCANNER_HOME/bin:$PATH

5. 添加 java 执行环境

exec /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.292.b10-0.el8_3.x86_64/bin/java \　　#原本是 exec "$java_cmd"

-Djava.awt.headless=true \

$SONAR_SCANNER_OPTS \

$SONAR_SCANNER_DEBUG_OPTS \

```
    -classpath "$jar_file" \
    -Dscanner.home="$sonar_scanner_home" \
    -Dproject.home="$project_home" \
    org.sonarsource.scanner.cli.Main "$@"
```

5. 项目下执行 sonar-scanner 扫描项目

sonar-project-.properties

```
sonar.projectName=loopoSonar                        项目名
sonar.projectKey=loopoSonarffff5325get6546fsfet     项目密钥
sonar.projectVersion=1.0                    项目版本
sonar.sources=.                             扫描路径
sonar.sourceEncoding=UTF-8                  文件编码
sonar.language=java                          扫描语言
sonar.scm.disabled=true
sonar.java.source=1.8
sonar.java.target=1.8
sonar.exclusions=**/test/**,**/target/**    排除目录
```

docker

Core

Docker Architecture

Docker Hub

- This is a cloud-based repository where Docker images are stored and shared.
- It serves a similar purpose to GitHub but specifically for Docker images, allowing developers to publish and manage their images easily.

Client-Server Architecture

- Server
  - The Docker daemon (or service) runs as a background process, managing Docker containers, images, networks, and volumes.
  - It listens for API requests and manages the execution of commands.
- Client
  - The Docker client is the command-line interface (CLI) that allows users to interact with the Docker daemon.
  - Users can send commands through the client, which communicates with the server via command line or REST API.
- Docker-in-Docker (dind)
  - This configuration allows you to run Docker commands inside a Docker container.
  - It is useful for testing and CI/CD pipelines, where you may need to build or run additional Docker containers from within a container.

Components

Images

- A Docker image is a read-only template used to create containers.
- It includes everything needed to run an application, such as the code, libraries, dependencies, and environment settings.
- For example, mysql:5.7 is an image for a specific version of MySQL.

Containers

- When a Docker image is executed, it creates a container, which is a running instance of that image.
- Containers operate independently, ensuring isolation and resource allocation (CPU, memory) without interfering with each other.

Data Volumes

A data volume is a persistent storage mechanism in Docker that allows data to be stored outside of containers.

It points to a specific directory on the host file system, enabling bidirectional modification of data that persists even if the container is deleted.

For example, data volumes are often found at /var/lib/docker/volumes/.

Comparison with Virtual Machines

Virtual Machines

A VM includes a full operating system (OS) along with the application, libraries, and dependencies.

Each VM runs its own kernel and requires significant system resources, leading to slower startup times and larger storage requirements.

VMs communicate with the host OS through a hypervisor, which allocates resources.

Containers

Containers share the host OS kernel but run in isolated environments, allowing for faster startup times (often just seconds) and reduced overhead.

This leads to more efficient resource usage. Docker containers are essentially processes running on the host OS.

兼容原理介绍

内核　　与硬件交互，提供操作硬件的指令

系统应用　封装内核指令为函数，便于程序员调用

用户程序　基于系统函数库实现功能（Ubuntu 和 CentOS 都是基于 Linux 内核，只是系统应用不同，提供的函数库有差异 ==> 用户程序在不同的系统上不能调用某些函数库）

Docker 如何解决兼容性问题的：函数库，依赖，配置，应用一起打包

将每个应用放到一个隔离容器去运行，避免相互干扰

Docker 如何解决不同系统环境的问题：Docker 将应用程序与所需要调用的系统（如 Ubuntu）函数库一起打包

Docker 运行到不同操作系统是，直接基于打包的库函数，借助于操作系统的 LInux 内核来运行

Docker 如何结局大型项目依赖关系复杂，不同组件依赖的兼容性问题：

Docker 允许开发中将　应用，依赖，函数库，配置一起打包，形成可移植镜像

Docker 应用运行在容器中，使用沙箱机制，相互隔离

Docker 如何解决开发，测试，生产环境有差异的问题：

Docker 镜像中包含完整的运行环境，包括系统函数库，仅依赖系统的 Linux 内核，因此可以在任意 Linux 操作系统上运行

介绍

DockerHub：一个 Docker 镜像的托管平台，称为 Docker Registry，类似 github

Docer 架构：Docker 是一个 CS 架构的程序，由两部分组成：

服务端（server）：Docker 守护进程，负责处理 Docker 指令，管理镜像，容器等

客户端（client）：通过命令或 RestAPI 向 Docker 服务端发送指令，可以在本地或远程向服务端发送指令

docker:dind　　需要在容器内再执行 docker 命令时使用

Configuration

Windows Installation

官方安装方式：https://docs.docker.com/engine/install/centos/　（Centos 8 使用 yum install docker -y 时，默认安装的是 podman-docker 软件）

systemctl start docker　启动 docker

docker -v

systemctl enable --now docker　　设置开机自启动

添加加速镜像文件：vi /etc/docker/daemon.json （ https://cr.console.aliyun.com/cn-hangzhou/instantces/mirrors ）

```
{
    "registry-mirrors": ["https://zydiol88.mirror.aliyuncs.com"]
    "insecure-registries": ["192.168.65.102:48083"]                    #非安全的镜像仓库地址，搭建本地镜像仓库使用
}
```

Commands

docker --version View version

--help Get help for any Docker command

docker info View docker state information.

Working with Docker Images

docker images

List all images

docker build -t <image-name>:<tag> <path-to-Dockerfile-directory>

Build an image from a Dockerfile

-t <image-name>:<tag>

Tag the image with a name and version.

--build-arg <ARG_NAME>=<value>

Pass build arguments to the image.

docker pull <image-name>:<tag>

Pull an image from a repository

Ensure Docker Desktop is Running

docker tag <source-image>:<tag> <target-image>:<new-tag>

Tag an image (rename or retag)

```
docker tag container-registry.oracle.com/db:23c myrepo/oracle23c:latest
```

docker push <image-name>:<tag>

Push an image to a repository

docker rmi <image-name>:<tag>

Remove an image

docker save -o <file-name>.tar <image-name>:<tag>

Save an image to a tar archive

docker load -i <file-name>.tar

Load an image from a tar archive

Working with Containers

docker ps View the status of all running container.

-a Display all containers, including stopped containers.

docker run <image-name>:<tag> <command>

Run a container from an image

```
docker run -d -p 9090:9090 -v /path/on/host/springboot-1.jar:/path/in/container/springboot-1.jar --name
springboot java:8 java -jar /path/in/container/springboot-1.jar
```

--name <container-name>

Name the container.

-d

Run the container in the background (detached mode).

-p <host-port>:<container-port>

Map host port to container port.

-v <host-path>:<container-path>

Mount a volume.

-e <ENV_VAR>=<value>

Set environment variables.

--link <container>:<alias>

Link another container.

--restart=always

Restart the container automatically.

docker start <container-name>

Start a stopped container

docker stop <container-name>

Stop a running container:

docker restart <container-name>

Restart a container

docker pause <container-name>

Pause a container

docker unpause <container-name>

Unpause a container

docker rm <container-name>

Remove a container

-f Force remove a running container.

docker inspect <container-name>

Inspect a container's details

docker logs <container-name>

View container logs

-f Follow the log output.

docker exec -it <container-name-or-id> <command>

Execute a command inside a running container

```
docker exec -it <container-name-or-id> bash
```

Use the docker exec command to open a terminal session inside the container:

-it Interactive terminal mode

docker cp <container-name>:/path/in/container /path/on/host

docker cp /path/on/host <container-name>:/path/in/container

Copy files between a container and the host

Working with Docker Volumes

docker volume create <volume-name>

Create a volume

docker volume inspect <volume-name>

Inspect a volume

docker volume ls

List all volumes

docker volume rm <volume-name>

Remove a volume

docker volume prune

Remove unused volumes

Working with Docker Networks

docker network create <network-name>

Create a network

docker network ls

List all networks

docker network rm <network-name>
Remove a network

docker inspect nginx-11　　查看容器信息
docker pause　　暂停（挂起进程，容器所占的内存暂存起来）
docker unpause
docker stop xxx　　停止容器 xxx（终止进程，容器所占的内存回收）
docker start xxx　　启动容器 xxx
docker restart xxx　　重启容器（不会影响容器文件）
docker ps　　查看所有运行的容器及状态【-a 展示所有容器，包含停止的容器】
docker logs xxx　　查看容器 xxx 运行日志 【-f 持续日志输出】
docker exec -it xxx bash　　进入容器执行命令，执行 bash 命令进入 linux 终端（查看当前镜像的文档，配置程序，使用 sed，不推荐在容器内修改）【-it 给当前进入的容器创建一个标准输入，输出终端，允许我们与容器交互】
docker rm -f xxx　　删除指定容器（包括容器在硬盘上的文件系统全部删除）【-f 强制删除运行中的容器】
docker cp　96f7f14e99ab:/www　/tmp/　　将容器内目录拷贝到主机，相反从主机拷贝到容器

数据卷：
docker volume create -name xxx　　创建一个数据卷 xxx 【-name xxx 指定数据卷名称，如果没有指定 name 使用随机的名称】
docker volume inspect xxx　　显示一个或多个数据卷 xxx 的信息（查看挂载点 Mountpoint，一般存储在目录_data 内）
docker volume ls　　列出所有数据卷
docker volume prune　　删除未使用的数据卷
docker volume rm xxx　　删除一个或多个指定的数据卷
网络
docker network create jenkins　　创建一个桥接网络
docker network ls　　显示网络列表
docker network rm xxx　　删除一个网路

docker run -d -p 9090:9090 -v /usr/springboot-1.jar:/usr/springboot-1.jar --name springboot java:8u111 java -jar /usr/springboot-1.jar　　docker 启动 jar

Dockerfile

FROM ubuntu:20.04
This command specifies the base image for your Docker image. It is typically the first instruction in a Dockerfile.
MAINTAINER <username or email>
Although deprecated in favor of the LABEL instruction, this command was used to define the maintainer of the image. (docker commit -a ...)
LABEL version="1.0" description="My application image"
This command adds metadata to your image. You can define multiple labels in a single instruction
RUN apt-get update && apt-get install -y curl (shell)　　RUN ["echo", "hello", "dd"] (exec)
This command executes commands in the shell inside the container and creates a new layer in the image.
It's often used for installing packages or setting up the environment
COPY ./src /app/src (shell)　　COPY ["index.html ", "/var/www/html "] (exec)
This command copies files or directories from your local filesystem into the Docker image. It preserves the file permissions:
ADD ./archive.tar.gz /app/

Similar to COPY, but with additional capabilities, such as automatically extracting compressed files.

It is recommended to use COPY unless you need the extra functionality

If the container directory does not exist, a new directory will be created automatically.

If it is a file, it will be overwritten directly if it already exists.

```
ADD src /usr/local
```

This command will add the content of src folder and not the folder itself

WORKDIR /app

This command sets the working directory for any subsequent commands. (e.g. RUN, CMD, ENTRYPOINT, ADD, COPY) (docker run -w <workdir>)

If the directory doesn't exist, Docker creates it

EXPOSE 8080

This command informs Docker that the container listens on the specified network ports at runtime.

It does not publish the ports; you must do that when running the container (e.g. docker run -p <port>)

CMD nginx -g daemon off (shell) CMD ["nginx", "-g" , "daemon off"] (exec) CMD ["-g" , "daemon off"] (Parameters provided to ENTRYPOINT)

This command specifies the default command to run when the container starts.

You can only have one CMD instruction in a Dockerfile. If there are multiple CMD instructions, only the last one will take effect

If you use the run command specified by docker run when starting the container, the CMD command will be overwritten.

ENTRYPOINT ["java", "-jar", "app.jar"] (exec) ENTRYPOINT nginx -g daemon off (shell)

Similar to CMD, this command defines the main command for the container, but it cannot be overridden when you run the container.

It is typically used for applications that require command-line arguments:

You can only have one ENTRYPOINT instruction in a Dockerfile. If there are multiple ENTRYPOINT instructions, only the last one will take effect

If you use the run command specified by docker run when starting the container, the ENTRYPOINT command will be overwritten.

VOLUME ["/data"]

This command creates a mount point with the specified path and marks it as holding externally mounted volumes from native host or other containers

ARG VERSION=1.0

This command defines a variable that users can pass at build-time to the builder with the docker build command.

It can be useful for setting dynamic build parameters:

Usage

```
# Use an official Java runtime as a parent image
FROM openjdk:11-jre-slim

# Set the working directory in the container
WORKDIR /app

# Copy the local JAR file into the container
COPY target/myapp.jar myapp.jar

# Expose the port that the application runs on
EXPOSE 8080

# Command to run the application
ENTRYPOINT ["java", "-jar", "myapp.jar"]
```

dockerCompose

The docker-compose command is used to define and run multi-container Docker applications.

It allows you to manage multiple services and their configurations in a single YAML file, called docker-compose.yml.
With docker-compose, you can start, stop, and manage multiple Docker containers at once.

Commands

docker-compose up

-f path/to/docker-compose.yml

Specify the configuration file path.

-d Stands for detached mode with the docker-compose up command.

This will start your containers and detach from the terminal, allowing the services to run in the background.

docker-compose down

Stop and remove containers

docker-compose ps

View running containers.

docker-compose logs -f 查看日志

docker-compose restart gateway userservice orderservice 重启除了 nacos 之外的服务（nacos 启动慢，其他服务注册不上终止了，需要重启）

docker tag nginx:latest 192.168.150.101:8080/nginx:1.0 打包一个新名称的镜像

docker push 192.168.150.101:8080/nginx:1.0 推送镜像

docker pull 192.168.150.101:8080/nginx:1.0 拉取镜像

私有仓库搭建

1. docker-compose.yml 部署私有仓库
2. 将仓库地址添加到 docker 服务的 daemon.json 文件中，信任地址

   vi etc/docker/daemon.json

   添加 "insecure-registries":["http://192.168.150.101:8080"]
3. 推送使用 docker push
4. 拉取使用 docker pull 命令

docker run -d --restart=always --name registry -p 5000:5000 -v registry-data:/var/lib/registry registry 私有仓库简易部署，没有图形界面

docker-compose.yml >> 私有仓库部署，带有图形界面（可以搭建 Shipyard）

```
version: '3.0'
services:
  registry:
    image: registry
    volumes:
      - ./registry-data:/var/lib/registry
  ui:
    image: joxit/docker-registry-ui:static
    ports:
      - 8080:80
    environment:
      - REGISTRY_TITLE=saidake 仓库
      - REGISTRY_URL=http://registry:5000
    depends_on:
      - registry
```

私有仓库 shipyard 搭建

官网安装手动：https://shipyard-project.com/manual-deployment/

rethinkdb	shipyard 数据库，nosql
microbox/etcd	服务注册，发现系统
shipyard/docker-proxy	docker API 代理，连接本地/var/run/docker.sock 代理用于让 swarm agent 连接 api 管理
swarm	官方管理 docker 集群工具
shipyard/shipyard	shipyard 前端

Configuration

docker-compose.yml

```
version: '3.8'   # Specify the Docker Compose file format version

services:   # Define the services that make up the application
  rabbitmq:   # Name of the RabbitMQ service
    image: rabbitmq:3-management   # Use RabbitMQ image with management plugin for easy monitoring
    container_name: rabbitmq   # Custom name for the RabbitMQ container
    ports:   # Expose ports for RabbitMQ
      - "5672:5672"   # Main RabbitMQ port for messaging
      - "15672:15672"   # Management interface port for monitoring RabbitMQ
    environment:   # Environment variables for configuring RabbitMQ
      RABBITMQ_DEFAULT_USER: guest   # Default username for RabbitMQ authentication
      RABBITMQ_DEFAULT_PASS: guest   # Default password for RabbitMQ authentication
      RABBITMQ_VHOST: /   # Default virtual host in RabbitMQ
      RABBITMQ_NODE_PORT: 5672   # Specify the port for RabbitMQ node communication
    volumes:   # Bind mount volumes to persist RabbitMQ data
      - rabbitmq_data:/var/lib/rabbitmq   # Persistent data storage for RabbitMQ
      - $PWD/mysql/data:/var/lib/mysql                                    #   $PWD is the current Directory
    mem_limit: 512M                                    #limit container memory

volumes:              # Define named volumes for persistent data storage
  rabbitmq_data:          # Named volume for RabbitMQ data, the colon is used to declare a named volume without additional options. Here's a breakdown of its significance.
```

harbor (Image Management)

安装

1. 下载压缩包：https://github.com/goharbor/harbor/releases
2. 解压缩：tar -xzvf harbor-offline-installer-v1.9.2.tgz -C /opt
3. 修改配置文件：vi harbor.yml

```
hostname: 192.168.66.102
http:
  port: 48083
#https      注释 https
```

4. 执行命令，拉取镜像，初始化安装：./prepare
5. 启动软件：./install.sh
6. 登录 harbor：http://sdk.vin:48083 admin = Harbor12345

docker-compose stop 停止（/usr/local/harbor 目录）

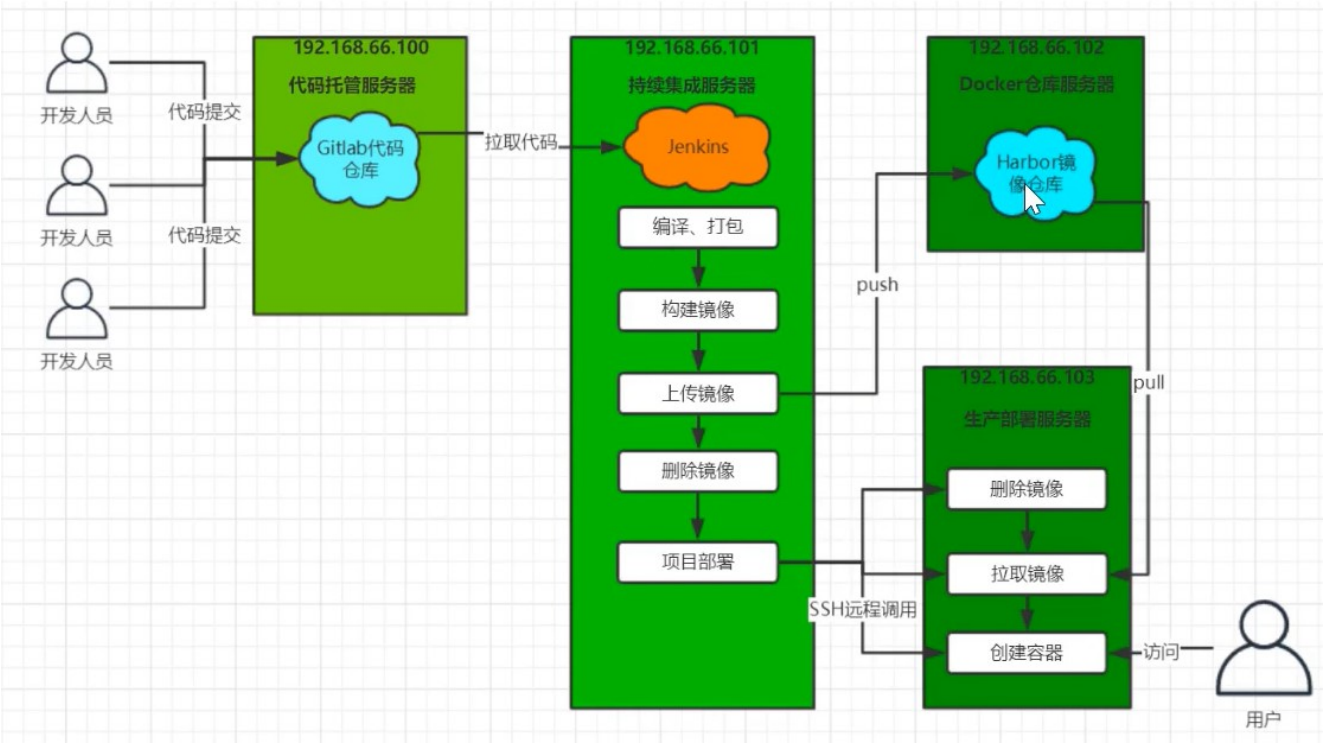
docker-compose start 启动

使用

1. 创建新用户

2. 新建私有项目，成员内添加访问用户【项目管理员：最高权限　维护人员：上传和下载和维护配置　开发人员：可以上传下载不能删除　访客：只能下载】

流程：



镜像管理

集成服务器：

1. 添加信任地址：vi /etc/docker/daemon.json

```
{
    "registry-mirrors": ["https://zydiol88.mirror.aliyuncs.com"]
    "insecure-registries": ["192.168.65.102:48083"]    自己的 ip
}
```

2. 重命名镜像：　docker tag eureka:v1 192.168.66.102:48083/xxpro/xxmyname:v1　　eureka:v1 原始镜像名，dockerfile-maven 打包后为 projectname:latest　　/xxpro/xxmyname:v1　项目名/最终镜像名

3. 远程登录镜像仓库服务器　docker login -u eric -p xxxpwd 192.168.66.102:48083

4. 推送镜像：docker push 192.168.66.102:48083/xxproname/xximgname:v1

生产服务器：

1. 添加信任地址：vi /etc/docker/daemon.json

```
{
    "registry-mirrors": ["https://zydiol88.mirror.aliyuncs.com"]
    "insecure-registries": ["192.168.65.102:48083"]
}
```

2. 远程登录镜像仓库服务器　docker login -u eric -p xxxpwd 192.168.66.102:48083

3. 下载镜像：docker pull 192.168.66.102:48083/xxproname/xximgname:v1

SeaLight
Concept
SeaLight Overview

SeaLights has built a comprehensive solution that integrates with your development pipeline and helps your team manage untested code in these areas primarily:

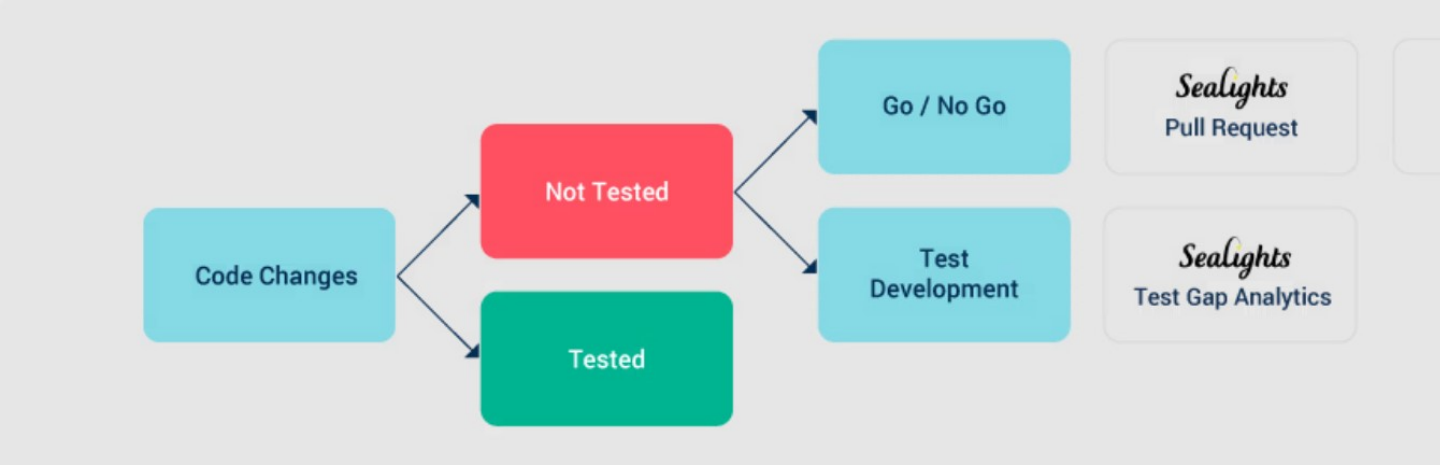
Pipeline checkpoints or go/no-go decision points.

1. Conforms to your quality policy (quality gates)
2. Pre-merge - integrate with your pull-request procedure
3. Post-merge - also integrate with your build-promotion processes

Test Assessment and Development

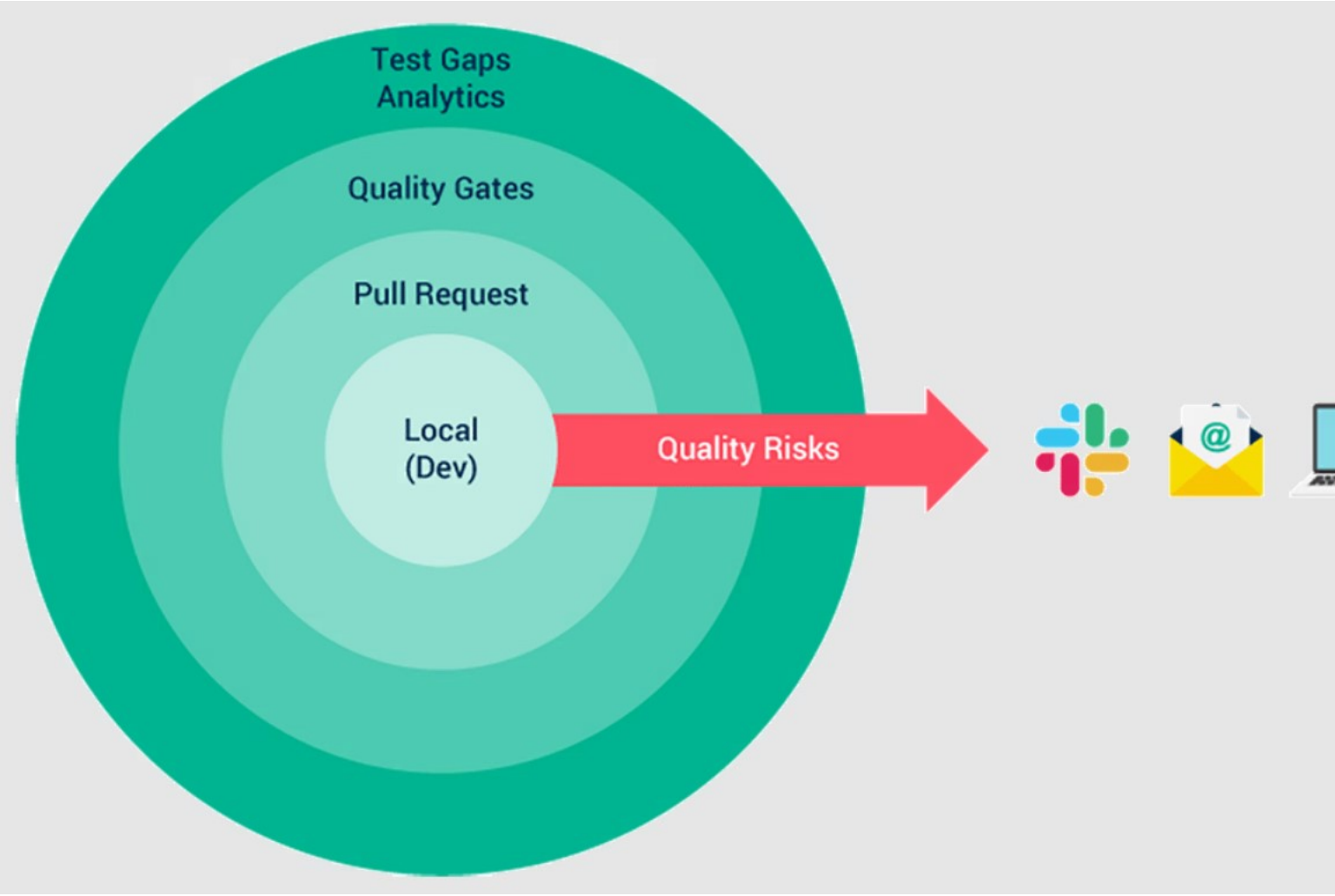
SeaLights will help your team identify:

1. What changes have been made
2. what has not been tested
3. Areas in which you need to add tests



Quality Approach -Multi-Layers

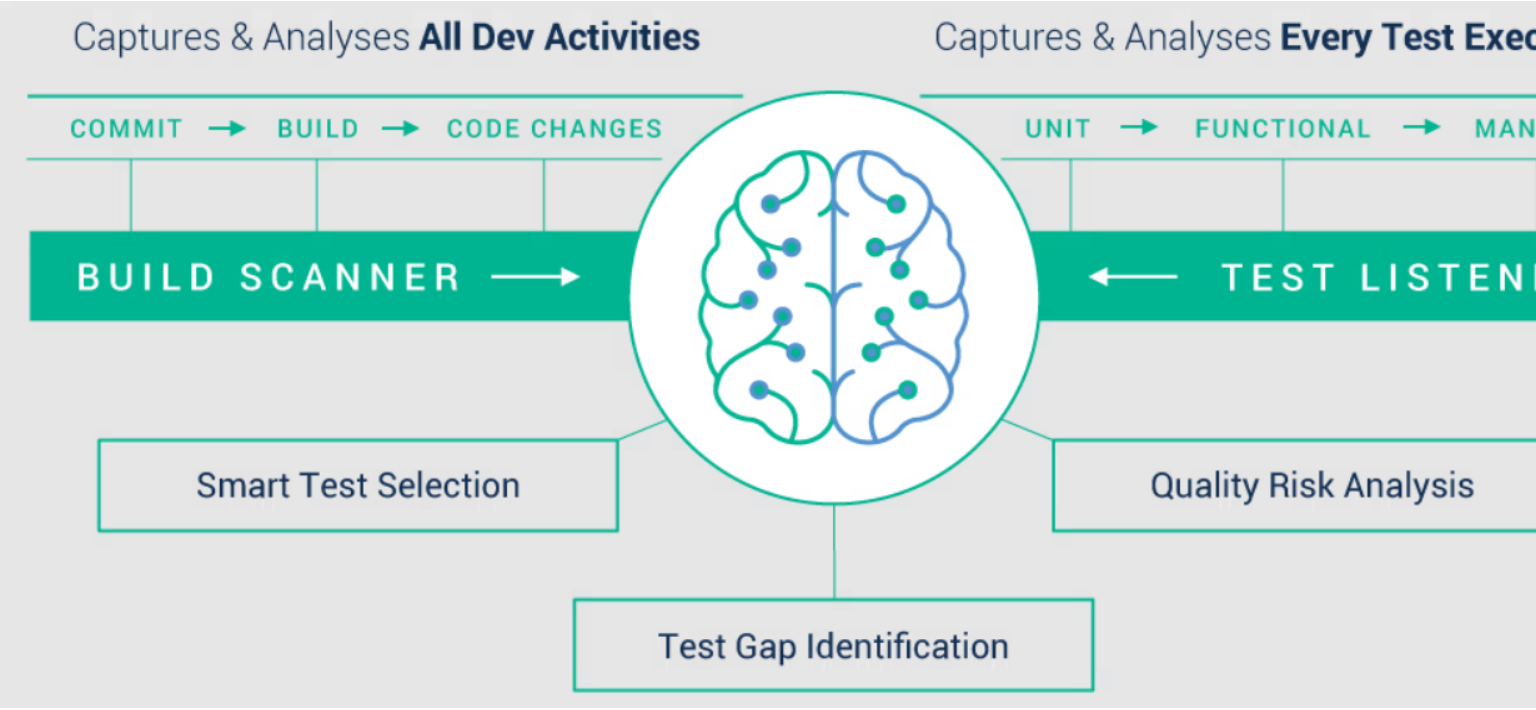
SeaLights Quality Intelligence Platform collects your code changes in each build,
which tests executed and what their code coverage,
calculates the quality gates based on your criteria given,
integrates into your pull request procedures and local development repositories and quickly communicates the level of QA exposure in the most recent build.



How does it work?

SeaLights employs two types of agents to achieve complete visibility of your software quality:

1. Build Scanner
   this agent scans all of your binaries and artifacts to create an intelligent mapping of your code and identified code changes.
2. Test Listener
   this agent runs along with your test management tools to identify which test footprints can be found on your code.



Preparation, Configuration, and Use

This outline presents all the major steps for preparation and configuring SeaLights for use in your organization.

Preparation

As you prepare to integrate SeaLights with your environment, you must make these preparations:.

1. The application build has been made ready for integration with SeaLights
2. Tests are available that can be run against the application
3. Connectivity is verified on the SeaLights dashboard for each machine on which the SeaLights agent has been installed.

Configure the build scanner

Next, you'll need to add code coverage by configuring the build scanner. This involves:.

1. Generating a token
2. Install and setup the appropriate agents and plug-ins on all participating machines.
3. Scan the application build.

Integrate Sealights with your build pipeline/process

Integrating SeaLights will require:

1. Configuring SeaLights to work with your CI/CD pipeline -OR- configuring SeaLights to work with your non-CI environment.
2. Capturing all key information about the applications and technical details for your environment.
3. Enabling user types and granting permissions to ensure the integration functions properly.

Add tests to the environment

To add tests, you'll need to:

1. Install the test listener(s)
2. Run the relevant tests

Analyze Quality Risks

The goal is to discover the quality risks that are lurking in your untested code.
Optimally , you want to find these risks upstream in your delivery pipeline—not in your production environment.
With SeaLights Release Quality Analytics, you can automate the detection of such exposures and better enforce high quality standards throughout your organization.
Learn more in Release Quality Analytics.

How to set up an application to report coverage to the Sealights dashboard

The following steps specify the necessary actions required to complete a standard onboarding of an application to report to the Sealights dashboard.

Sealights implementation - a Step by Step guide:

Step 1 - Getting Started

1. Users that require access are specified by the customer, then created by Sealights.
2. New user logs in for the first time using given credentials & sets a new password
3. From the Sealights dashboard, create an application token (Settings →Agent Token)
    a. This token should be saved in a file (default: sltoken.txt), saved in the root workspace directory

Step 2 - Running a Sealights cycle - Build Scan

1. Create a session ID - (Java,Node.JS,.NET,Python)
    a. Requires an application name, branch, build number & package names to be included in the scan.
    b. This step runs every time a new build is reported to the dashboard
2. Scan the binary files (Java, Node.JS,.NET,Python)
    a. Requires permissions on the server where artifacts are built & unit tests are run

Step 3 - Running a Sealights cycle - Test Listener

1. Configure unit tests to run with our test listener (Java - Maven/Gradle, Node.JS,.NET, Python)
    a. Method of configuration varies in accordance to how tests are run - executed by plugin/manually
2. Deploy the packaged application onto the application server
3. Configure application server to go up with our Test listener & ensure the test listener is started alongside the application server.(Java, Node.S,.NET,Python)
4. Start an execution, specify a test stage name.(Java,Node.JS,.NET, Python)
5. Run functiona/manual tests
6. End the execution of the test stage.(Java,Node.JS,.NET, Python)

How to setup the build scanner?

In order to setup a build you need to perform the following steps:

Generate a token which will be used with our agents.

- Generating a token

Depending on the architecture follow the following steps:

Java using Maven with Jenkins plugin

- SeaLights Jenkins plugin - Installation and setup.
- SeaLights Jenkins plugin - Generating a session ID
- SeaLights Jenkins plugin - Scanning Builds and running Unit Tests using Maven

Java using Gradle

- Scanning Builds and Tests using Gradle plugin

Java Command line

Download the java agent files to be used to scan the build

- Downloading the java agent files

Generate a session ID which will contain the common configuration details for the complete build and testing aycle

- Using Java Agents - Generating a session ID

Scan your build artifacts and provide the SeaLights server the needed information regarding it.

- Using Java Agents - Scanning a build

Node.js

Download the Node.js agent files to be used to scan the build

- Downloading the Node.js agent file

Generate a session ID which will contain the common configuration details for the complete build and testing aycle

- Using Node.,js Agent - Generating a session ID

Scan your build artifacts and provide the SeaLights server the needed information regarding it

- Using Node.,jsAgents - Scanning a build

Command
Build Scanner

Create a session ID

java -jar sl-build-scanner.jar

-config

{ -token <string> | -tokenfile <file-path>] Access token generated from the SeaLights server

-appname <app-name>

-branchname <branch-name>

-buildname <build-name> The build label of the current build

-pi <arg> Comma-separated list of packages to include in scan
Supports wildcards(* = any string, ? = any character). For example: "com.example.* ,io.*.demo, com.?ello.world"

[-pe <arg>] Comma-separated list of packages to exclude in scan

[-buildsessionid <arg>] Optional:A user provided session ID.(This ID must be under 40 characters length).

[-proxy <arg>] Address of proxy to run connection through

[-buildsessionidfile <arg>] Path to a file to save the build session id in. Default: buildsessionId.txt

LightSpeed
uDeploy

LSC (Lightspeed - Classical)

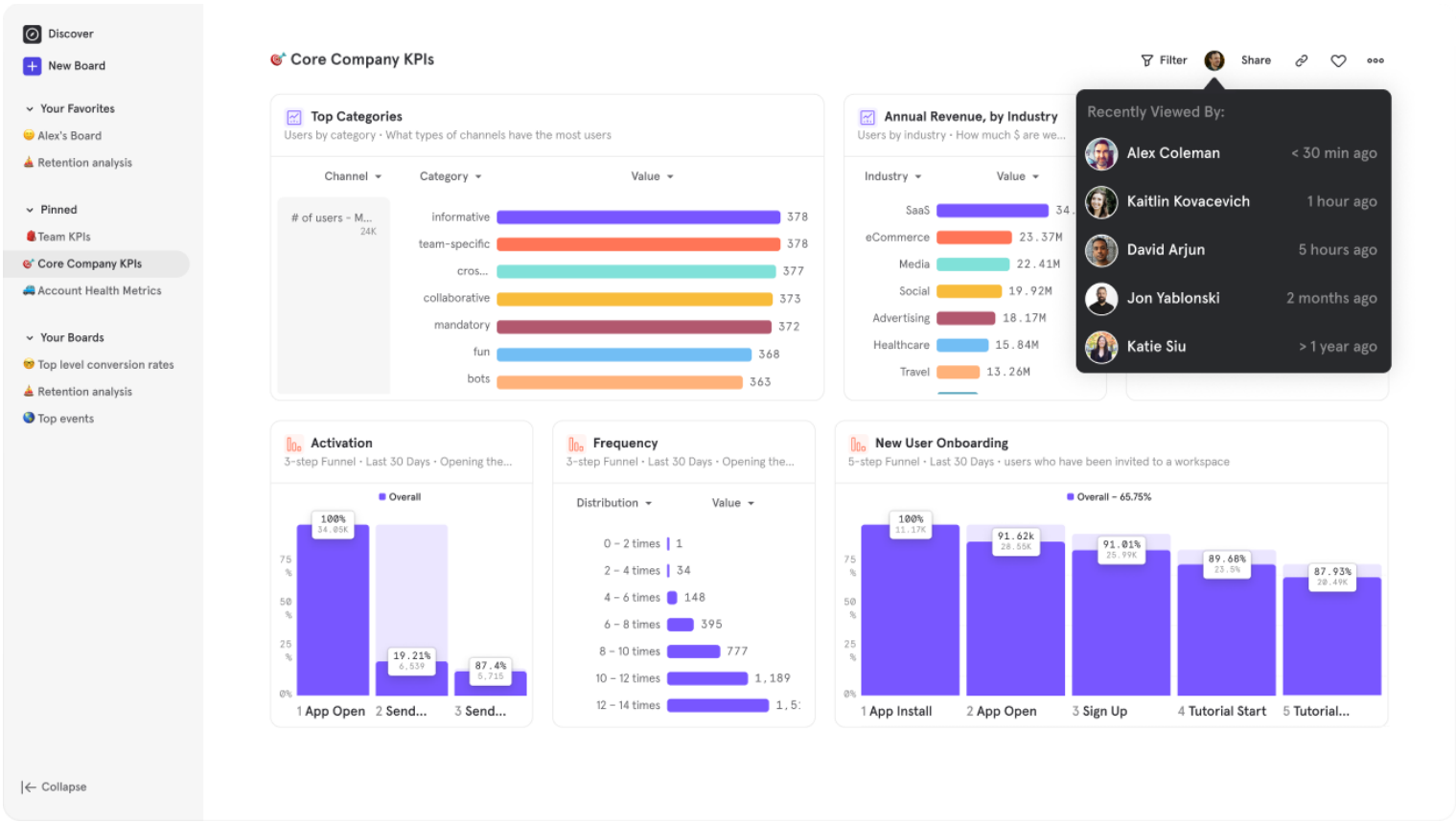
deploy with Jenkins and Udeploy.

harness

LSE (Lightspeed - Enterprise)

Harness

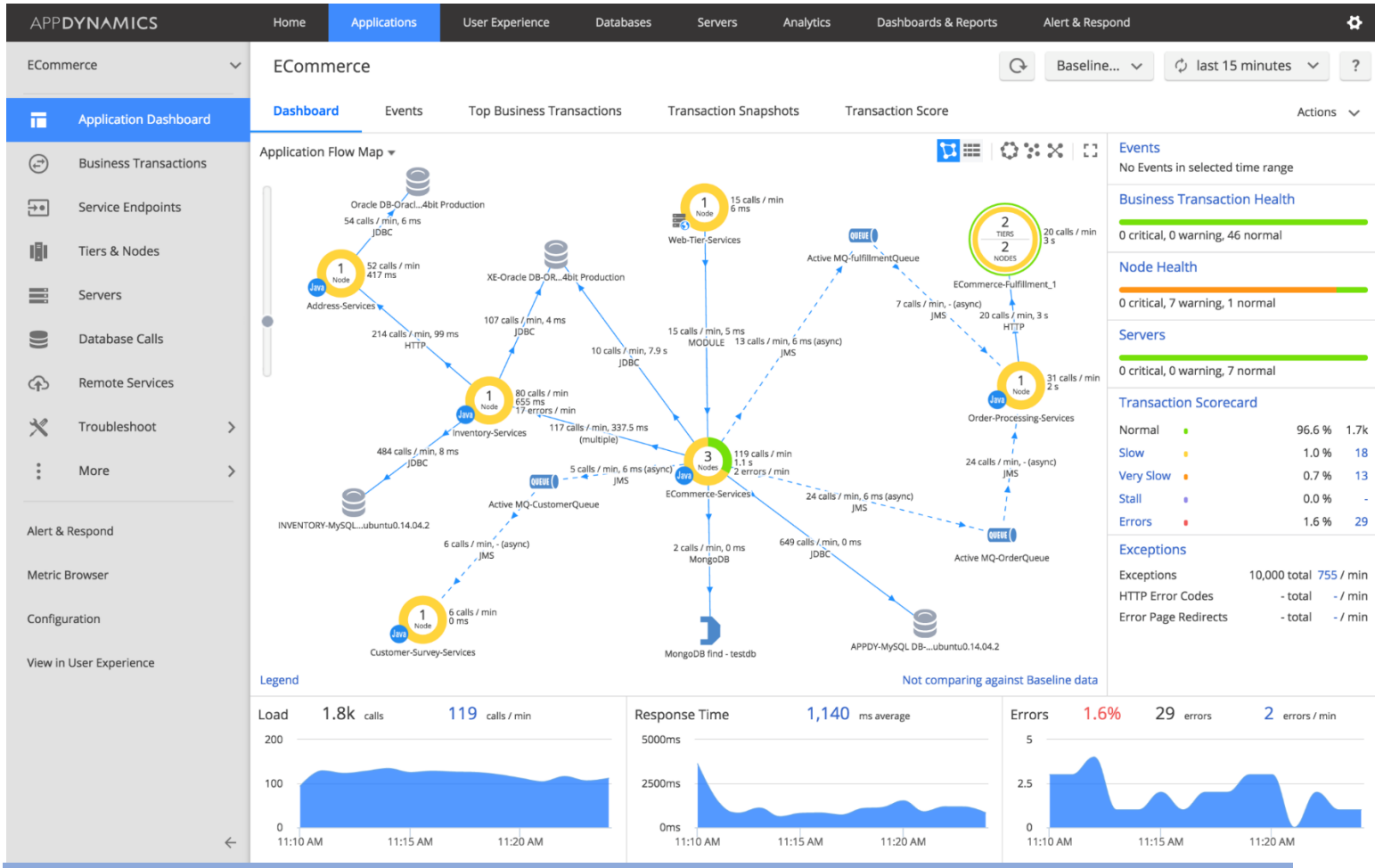
Mixpanel



Appdynamic

AppDynamics is focused on application performance management (APM).

It provides tools for monitoring the performance of applications, identifying bottlenecks, and diagnosing issues to ensure the application runs smoothly and efficiently.



Kubernetes

Core

Kubernetes vs. Docker

Docker is a containerization platform that packages applications and their dependencies into containers.
It provides a lightweight, isolated environment for applications to run consistently across different systems.
Kubernetes is a container orchestration platform designed to automate deployment, scaling, and management of containerized applications. It groups containers into logical units for easy management and discovery.
Key differences:

Scope:

Docker focuses on individual containers, while Kubernetes manages clusters of containers.

Functionality:

Docker is a tool for creating and running containers, while Kubernetes is a platform for orchestrating and managing containerized applications.

Features:

Kubernetes provides additional features like service discovery, load balancing, storage orchestration, and automated rollouts.

Functionality

- **服务发现和负载均衡**

Kubernetes 可以使用 DNS 名称或自己的 IP 地址来暴露容器。 如果进入容器的流量很大， Kubernetes 可以负载均衡并分配网络流量，从而使部署稳定。

- **存储编排**

Kubernetes 允许你自动挂载你选择的存储系统，例如本地存储、公共云提供商等。

- **自动部署和回滚**

你可以使用 Kubernetes 描述已部署容器的所需状态， 它可以以受控的速率将实际状态更改为期望状态。 例如，你可以自动化 Kubernetes 来为你的部署创建新容器， 删除现有容器并将它们的所有资源用于新容器。

- **自动完成装箱计算**

你为 Kubernetes 提供许多节点组成的集群，在这个集群上运行容器化的任务。 你告诉 Kubernetes 每个容器需要多少 CPU 和内存 (RAM)。 Kubernetes 可以将这些容器按实际情况调度到你的节点上，以最佳方式利用你的资源。

- **自我修复**

Kubernetes 将重新启动失败的容器、替换容器、杀死不响应用户定义的运行状况检查的容器， 并且在准备好服务之前不将其通告给客户端。

- **密钥与配置管理**

Kubernetes 允许你存储和管理敏感信息，例如密码、OAuth 令牌和 SSH 密钥。 你可以在不重建容器镜像的情况下部署和更新密钥和应用程序配置，也无需在堆栈配置中暴露密钥。

- **批处理执行** 除了服务外，Kubernetes 还可以管理你的批处理和 CI（持续集成）工作负载，如有需要，可以替换失败的容器。
- **水平扩缩** 使用简单的命令、用户界面或根据 CPU 使用率自动对你的应用进行扩缩。
- **IPv4/IPv6 双栈** 为 Pod（容器组）和 Service（服务）分配 IPv4 和 IPv6 地址。
- **为可扩展性设计** 在不改变上游源代码的情况下为你的 Kubernetes 集群添加功能。

Configuration

KUBECONFIG

If you have a single configuration file (usually at ~/.kube/config), it's automatically used by kubectl.
However, if you want to specify a custom path, set the KUBECONFIG environment variable.
On Linux/macOS:

```
export KUBECONFIG=/path/to/config1:/path/to/config2
```

On Windows:

```
$env:KUBECONFIG="C:\path\to\config1;C:\path\to\config2"
```

~/.kube/config

```
apiVersion: v1
kind: Config
clusters:
- name: my-cluster
  cluster:
    server: https://my-cluster-server:6443              # API server URL
    certificate-authority: /path/to/ca.crt              # Path to the CA certificate
    insecure-skip-tls-verify: true                      # Skip tls verify if you don't have a CA file
    certificate-authority-data: <Base64-encoded-CA-certificate>
                                                        # Use the Certificate Authority (CA) certificate data in Base64-encoded forma
users:
- name: my-user
  user:
    client-certificate: /path/to/client.crt             # Path to client certificate
    client-key: /path/to/client.key                     # Path to client key
    exec:       # Use an external command to retrieve authentication credentials dynamically, rather than embedding a static token or certificate ( client-certificate, client-key ).
      apiVersion: "client.authentication.k8s.io/v1"     # Required API version
      command: "aws-iam-authenticator"                  # Command to execute
      args:                                             # Optional: command arguments
```

```
      - "token"
      - "-i"
      - "my-cluster-id"
    env:                                    # Optional: environment variables for the command
      - name: "AWS_REGION"
        value: "us-west-2"
    installHint: "Please install aws-iam-authenticator."   # Provide an informative hint for users on how to install or configure the executable command if it is missing on their system.

contexts:
- name: my-context
  context:
    cluster: my-cluster
    user: my-user

current-context: my-context                 # Default context to use
```

Plugins

kubectl-oidc_login

brew install int128/kubelogin/kubelogin

Commands
kubectl

The kubectl command-line tool allows you to interact with a Kubernetes cluster, managing resources, pods, deployments, services, and more.

Here's a quick guide to getting started and using some common kubectl commands.

Usage

kubectl [command] [type] [name] [flags]

Basic Commands

kubectl cluster-info

Display cluster info

kubectl get events

Check recent events in the cluster

kubectl config view

Run the following command to verify your KUBECONFIG settings

kubectl config get-contexts

List available contexts

kubectl config use-context <context-name>

Switch to a different context

kubectl get nodes

List nodes in the cluster

kubectl describe node <node-name>

Describe a specific node

kubectl get namespaces

List namespaces

kubectl create namespace <namespace>

Create a new namespace

kubectl delete namespace <namespace>

Delete a namespace

Working with Pods

kubectl get pods
    List all pods
    -n <namespace> List pods in a specific namespace

kubectl get pod <pod-name> -o yaml
    Get details about a specific pod

kubectl describe pod <pod-name>
    Show detailed information about a pod

kubectl delete pod <pod-name>
    Delete a pod
    -n <namespace> Delete a pod in a specific namespace

kubectl exec -it <pod-name> -- /bin/bash
    Execute a command in a running pod

kubectl logs <pod-name>
    Show logs of a pod
    -c <container-name> Check logs for a specific container

Managing Deployments

kubectl create deployment <name> --image=<image>
    Create a deployment

kubectl get deployments
    List all deployments

kubectl describe deployment <deployment-name>
    Describe a deployment

kubectl delete deployment <deployment-name>
    Delete a deployment

kubectl scale deployment <name> --replicas=<number>
    Scale the deployment

kubectl set image deployment/<name> <container>=<new-image>
    Update container image

kubectl rollout restart deployment <deployment-name>
    Restart a Deployment

kubectl get all
    List all resources in the default namespace
    -n <namespace> List all resources in a specific namespace

Using Services

kubectl get services
    List all services

kubectl describe service <service-name>
    Describe a specific service

kubectl port-forward <pod-name> <local-port>:<pod-port>
    Forward a local port to a pod's port

ConfigMaps and Secrets

kubectl create configmap <name> --from-literal=<key>=<value>
    Create ConfigMap from literal

kubectl create configmap <name> --from-file=<path-to-file>
    Create ConfigMap from file

kubectl get configmap
    List ConfigMaps

kubectl describe configmap <name>

Describe a ConfigMap

kubectl create secret generic <name> --from-literal=<key>=<value>

Create Secret from literal

kubectl get secrets

List secrets

kubectl describe secret <name>

Describe a secret

Apply and Edit Configurations

kubectl apply -f <file.yaml>

Apply a configuration file (Create a deployment from a YAML file).

kubectl delete -f <file.yaml>

Delete resources in a configuration file

kubectl edit deployment <deployment-name>

Edit a deployment configuration

potman

Podman vs. Docker

Daemon-less Architecture:

Podman's architecture eliminates the need for a central daemon, offering more flexibility and security.

Rootless Container Support:

Podman provides better support for running containers without root privileges, reducing potential security risks.

Pods Feature:

Podman natively supports the concept of pods, which is a group of containers sharing the same resources, similar to Kubernetes.

Command Compatibility:

While Podman aims to be Docker-compatible, some advanced features may behave differently or require adjustments.

Using Podman as a Docker Alternative

Podman doesn't require a central daemon like Docker. However, to make it work seamlessly with Docker commands, you can follow these steps:

Podman doesn't support amd64 archi image in macos. (colima)

Set Up Podman to Work Like Docker:

Podman can be used as a Docker replacement by setting up a compatibility service with Docker's socket.

```
podman machine init
podman machine start
```

Set Up the Podman Machine:

Initialize the Podman machine (a lightweight virtual machine that runs containers).

```
podman machine init
```

Start the Podman machine.

```
podman machine start
```

Configure the Docker-Compatible Socket:

Podman provides a Docker-compatible socket to enable Docker CLI support.

To enable this, use:

```
podman system service --time=0 &
```

This starts the Podman service and allows Docker-compatible commands to be used.

Set Up the Alias for Docker Compatibility

Set up an alias to make docker commands work with Podman.

Add the this line to your shell configuration file (e.g., ~/.zshrc or ~/.bash_profile) to make it persistent.

```
alias docker=podman
unalias docker
```

colima

Use Docker with Colima

Once Colima is running, you can use Docker commands as usual.

Colima integrates with the Docker CLI, so commands like docker ps, docker run, and others will work as expected:

```
docker run hello-world
```

Commands

`colima start`

To start Colima with Docker

By default, this will set up a virtual machine with Docker.

Colima automatically configures the Docker CLI to use its environment.

If you need to adjust resources (e.g., CPUs, memory) after starting Colima, first stop it, then restart with new options:

`colima start --memory 4 --cpus 2`

Memory and CPU allocation

`colima start --kubernetes`

Kubernetes support

`colima start --mount ~/my-directory`

Mount a directory

`colima start --kubernetes`

To use Kubernetes with Colima, start Colima with the --kubernetes option

colima status

Check Colima's current status

colima delete

Remove the Colima instance and its associated data

sdkman

Configuration

Install SDKMAN

Commands

sdk version

sdk install java

To install the latest version of an SDK

sdk install java 17.0.1-tem

To install a specific version, specify the version number.

sdk uninstall java 17.0.1-tem

To remove a version.

sdk list java

To list all installed versions of an SDK (e.g., Java):

sdk use java 11.0.11-open

To switch to a specific version temporarily in a session

sdk default java 11.0.11-open

To set a specific version as the default:

sdk selfupdate

To update SDKMAN! itself

sdk upgrade

To check for updates to all installed SDKs.

Config / Work Environment

LocalStack

Installation

Create a Docker Compose file for LocalStack

In your project directory, create a file called docker-compose.yml with the following content:

```
version: '3.8'

services:
  localstack:
    image: localstack/localstack:latest
    container_name: localstack
    ports:
      - "4566:4566"   # Main entry point for all services
      - "4571:4571"   # Optional services like Lambda, DynamoDB
    environment:
      - SERVICES=s3,ec2,lambda,dynamodb   # Define the AWS services you want to simulate
      - DEFAULT_REGION=us-east-1
      - EDGE_PORT=4566
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"  # Enable Docker-in-Docker
```

Start and Stop LocalStack

Run Docker Compose to start LocalStack:

This command will pull the LocalStack image, start the services you specified (s3, ec2, lambda, dynamodb), and map them to ports on your local machine.

```
docker-compose up
docker-compose down
```

Access LocalStack

LocalStack's main endpoint will be available at http://localhost:4566.

Use AWS CLI or SDKs with --endpoint-url=http://localhost:4566 to interact with LocalStack's simulated services.

awscli

Mac Installation Introduction

The "examples" directory has been installed to:

/usr/local/share/awscli/examples

zsh completions and functions have been installed to:

/usr/local/share/zsh/site-functions

Installation

Configuration

Installation

If you haven't installed AWS CLI yet, you can install it to test LocalStack services.

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

or

```
sudo apt install awscli
aws configure
```

You can use dummy values for the Access Key, Secret Key, and set the region to something like us-east-1.
When interacting with LocalStack services, always specify the --endpoint-url flag:

```
aws s3 ls --endpoint-url=http://localhost:4566
```

Global Environment Variables

export AWS_ENDPOINT_URL=http://localhost:4566

~/.aws/credentials

You can create multiple named profiles in your ~/.aws/credentials file and specify which profile to use:
When you execute AWS CLI commands over SSH on a remote server,
the AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY environment variables you set in your script need to match the credentials stored in the ~/.aws/credentials file on that server
if you intend to use those credentials for authentication.

```
[profile_name]
aws_access_key_id=your_access_key
aws_secret_access_key=your_secret_key
```

~/.aws/config

[default]
region = us-east-1

[profile localstack]
region = us-east-1
# default endpoint
s3 =
    endpoint_url = http://localhost:4566

Optional Tools

gimme-aws-creds

After configuring the ~/.okta_aws_login_config file, the gimme-aws-creds command will automatically read the configuration file.

gimme-aws-creds
    Authenticate with your preferred identity provider (e.g., Okta):
gimme-aws-creds --profile "gimme-creds"
    Get temporary credentials

Command	
aws --version	View version
aws configure list	View configuration list

aws configure set region <region_name> Set aws region

```
aws configure set region us-west-2
```

aws configure --profile <profile-name> Set a profile name for the current configuration, and fill out the configuration form.

EC2

aws ec2 create-key-pair

--key-name <key-pair-name>

Name of the key pair to create.

--query 'KeyMaterial'

This specifies that you only want to extract the value associated with the KeyMaterial key from the JSON response returned by the create-key-pair command.

--output text > <key-pair-name>.pem

Creates a new key pair that you can use to connect to your instance.

KeyMaterial:

This key contains the private key material for the key pair in PEM format.

When you create a key pair using the AWS EC2 CLI, AWS returns a JSON response containing various details about the key pair,

including:

KeyName	The name of the key pair.
KeyFingerprint	The fingerprint of the key.
KeyMaterial	The actual private key that you will use to connect to your EC2 instances.

aws ec2 delete-key-pair

--key-name <key-pair-name>

Irreversible Action:

Deleting a key pair is irreversible. Once deleted, you cannot recover it, and any instances that were using that key pair will no longer be accessible using that key.

No Impact on Running Instances:

Deleting a key pair does not terminate or affect any running EC2 instances.

However, you won't be able to connect to those instances using that key pair anymore.

Check for Dependencies:

Make sure the key pair isn't being used by any running instances before deleting it.

aws ec2 describe-images

--image-ids

Specifies one or more AMI IDs to describe. If you provide this option, only the specified AMIs will be described.

--owners

Filters the results to only include AMIs owned by specific AWS accounts.

Common values include self for your own AMIs or amazon for public AMIs provided by AWS.

--filters

Allows you to apply various filters to narrow down the results.

For example, you can filter by name, state, or architecture.

--query

Uses JMESPath to format the output. This is useful for extracting specific fields from the response.

--output

Specifies the output format (e.g., json, table, text).

aws ec2 describe-images --owners self

aws ec2 describe-images --owners amazon --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"

aws ec2 describe-images --owners amazon --query "Images[*].[ImageId, Name]" --output table

aws ec2 run-instances

--image-id <ami-id> ID of the AMI to use.
--instance-type <instance-type> Type of instance (e.g., t2.micro, m5.large).
--key-name <key-pair-name> Name of the key pair for SSH access.
--security-group-ids <group-id> (Optional) Security group ID(s) to assign to the instance.
--subnet-id <subnet-id> (Optional) Subnet ID for launching the instance in a specific VPC.
This command launches a new EC2 instance using the specified AMI ID, instance type, and key pair.
Note: LocalStack does not provide actual AMIs or security groups

aws ec2 describe-instances
--instance-ids <instance-id> (Optional) Specify one or more instance IDs to filter results.
--query Use this option to filter the output with JMESPath syntax.
Lists information about your EC2 instances.

aws ec2 stop-instances
--instance-ids <instance-id-1> <instance-id-2> ID of the instance you want to stop.
Stops an EC2 instance.

aws ec2 start-instances
--instance-ids <instance-id> ID of the instance you want to start.
Starts a previously stopped EC2 instance.

aws ec2 terminate-instances
--instance-ids <instance-id-1> <instance-id-2> ID of the instance you want to terminate.
Terminates an EC2 instance.

aws ec2 reboot-instances
--instance-ids <instance-id> ID of the instance you want to reboot.
Reboots an EC2 instance without stopping it.

aws ec2 describe-key-pairs
Lists all key pairs in your account.

aws ec2 create-security-group
--group-name <group-name> Name of the security group.
--description "<description>" Description of the security group.
Creates a new security group.

aws ec2 authorize-security-group-ingress
--group-id <security-group-id> ID of the security group to modify.
--protocol tcp Protocol (e.g., tcp, udp).
--port 22 Port number or range to allow.
--cidr 0.0.0.0/0 CIDR block to allow traffic from.
Adds inbound rules to a security group.

aws ec2 describe-security-groups
Lists all security groups and their rules.

aws ec2 describe-instances
--instance-ids <instance-id>
--query 'Reservations[*].Instances[*].PublicIpAddress'
--output text
Retrieves the public IP address of a specific instance.

Lambda

aws lambda create-function
--function-name MyLambdaFunction
--zip-file fileb://path/to/your/package.zip

--zip-file fileb://build/libs/my-lambda-app-0.0.1-SNAPSHOT.jar
--handler com.example.MyLambdaHandler
--runtime java11
--role arn:aws:iam::123456789012:role/service-role/MyLambdaRole

You can also deploy your Lambda function using the AWS CLI. Use the following command:
Make sure to replace path/to/your/package.zip, com.example.MyLambdaHandler, and the role ARN with your actual values.

The handler is always org.springframework.cloud.function.adapter.aws.FunctionInvoker for AWS Lambda integration with Spring Cloud Function.
runtime should match the version of Java your Spring Boot app uses (e.g., java11).

aws lambda invoke
--function-name my-lambda-function
--payload file://input.json \
--payload '{"functionName": "greet", "data": "Alice"}'
--environment Variables="{VAR1=value1,VAR2=value2}"
output.json
--query 'Payload'
--output text

--function-name: The name of your Lambda function.
--payload: The input for the Lambda function, specified as a JSON file. Use the file:// prefix to read from a local file.
output.json: The file where the response from the Lambda function will be saved.
--query 'Payload': This specifies that you want to extract the Payload field from the command's response, which contains the output of your Lambda function.
--output text: This formats the output to plain text for easier reading in the console.

aws lambda delete-function
--function-name <function-name>
--region <region>
Delete a Lambda function:

aws lambda update-function-code
--function-name <function-name>
--zip-file fileb://<path-to-zip-file>
--region <region>
Update Lambda function code
You can update a Lambda function's code by uploading a ZIP file with the updated code

aws lambda list-functions
--region <region>
List all Lambda functions:
To list all Lambda functions in your AWS account for a specific region, run:

aws lambda get-function

--function-name <function-name>

--region <region>

Get details about a specific Lambda function:

To get detailed information about a specific Lambda function:

aws lambda invoke

--function-name <function-name>

--payload '<json-payload>'

--region <region>

output.txt

Invoke a Lambda function:

You can invoke a Lambda function directly from the CLI and pass a JSON payload to the function.

Use the following command to invoke a function and get the response:

aws logs filter-log-events

--log-group-name /aws/lambda/<function-name>

--region <region>

View recent logs for a Lambda function:

To see the logs from a Lambda function, you can use AWS CloudWatch Logs. First, you need the log group and log stream names.

By default, Lambda logs are stored in CloudWatch Logs under /aws/lambda/<function-name>.

aws lambda get-function-configuration

--function-name <function-name>

--region <region>

Check the function's configuration:

To check or retrieve the configuration of a Lambda function, such as the runtime, timeout, and memory size:

aws lambda get-policy

--function-name <function-name>

--region <region>

Check Lambda policies (to view resource-based policies that grant permissions to your function):

IAM

Global

aws iam get-account-summary

Returns a summary of IAM resources in your account, such as the number of users, groups, roles, and policies.

aws iam simulate-principal-policy --policy-source-arn <user-or-role-arn> --action-names <actions>

Simulates how the specified IAM policy or policies affect a list of API actions for a particular user or role.

aws iam tag-user

--user-name <username>

--tags Key=Department,Value=IT

Adds tags to an IAM resource.

aws iam list-user-tags

--user-name <username>

List tags for a resource

aws iam untag-user

--user-name <username>

--tag-keys Department

Remove tags from a resource

User Management

aws iam create-user

--user-name <username>

This creates a new IAM user.

aws iam list-users

Lists all IAM users in your AWS account.

aws iam attach-user-policy

--user-name <username>

--policy-arn <policy-arn>

Attach a policy to a user

Attaches an existing policy (e.g., AdministratorAccess) to the specified user.

aws iam delete-user

--user-name <username>

Deletes the specified IAM user.

Group Management

aws iam create-group

--group-name <groupname>

Creates a new IAM group.

aws iam list-groups

Lists all IAM groups in your AWS account.

aws iam add-user-to-group

--group-name <groupname>

--user-name <username>

Adds an IAM user to a specific group.

aws iam remove-user-from-group

--group-name <groupname>

--user-name <username>

Removes a user from a group.

Policy Management

aws iam create-policy

--policy-name <policyname>

--policy-document file://policy.json

Creates a new IAM policy from a JSON document.

aws iam list-policies

Lists all policies in your AWS account.

aws iam attach-group-policy

--group-name <groupname>

--policy-arn <policy-arn>

Attaches a policy to a specified IAM group.

A policy ARN (Amazon Resource Name) is a unique identifier for an AWS IAM policy.

When you attach a policy to a user, role, or group, you reference the policy using its ARN to give specific permissions to access AWS resources.

aws iam detach-user-policy

--user-name <username>

--policy-arn <policy-arn>

aws iam detach-role-policy

--role-name <rolename>

--policy-arn <policy-arn>

Detaches a policy from a user or role.

aws iam delete-policy

--policy-arn <policy-arn>

Deletes a custom IAM policy.

Role Management

aws iam create-role --role-name <rolename> --assume-role-policy-document file://policy.json

Create a new IAM role

This creates a new IAM role. The --assume-role-policy-document argument specifies a JSON file that defines the role's trust relationship.

aws iam list-roles

Lists all IAM roles.

aws iam attach-role-policy --role-name <rolename> --policy-arn <policy-arn>

Attaches a managed policy to an IAM role.

aws iam delete-role --role-name <rolename>

Deletes a specified IAM role.

Access Key Management

aws iam create-access-key --user-name <username>

Creates a new access key for the specified IAM user.

aws iam list-access-keys --user-name <username>

Lists the access keys associated with a user.

aws iam delete-access-key --access-key-id <access-key-id> --user-name <username>

Deletes the specified access key.

Password Management

aws iam create-login-profile --user-name <username> --password <password>

This command sets a password for an IAM user to allow them to log into the AWS Management Console.

aws iam delete-login-profile --user-name <username>

Deletes the password associated with the user.

Dynamodb

aws dynamodb list-tables

Lists all tables in your AWS account's DynamoDB.

aws dynamodb create-table \

--table-name <table-name> \

--attribute-definitions AttributeName=<attribute-name>,AttributeType=<type> \

--key-schema AttributeName=<attribute-name>,KeyType=<type> \

--provisioned-throughput ReadCapacityUnits=<value>,WriteCapacityUnits=<value>

Creates a new DynamoDB table with specified attributes, key schema, and provisioned throughput settings.

aws dynamodb update-table \

--table-name <table-name> \

--provisioned-throughput ReadCapacityUnits=<value>,WriteCapacityUnits=<value>

Modifies the settings for an existing DynamoDB table, such as throughput or global secondary indexes.

aws dynamodb delete-table \

--table-name <table-name>

Deletes a DynamoDB table and all of its data.

```
aws dynamodb put-item \
    --table-name <table-name> \
    --item '{"<attribute-name>": {"S": "<string-value>"}}'
```
Inserts a new item into a DynamoDB table.

```
aws dynamodb get-item \
    --table-name <table-name> \
    --key '{"<attribute-name>": {"S": "<key-value>"}}'
```
Retrieves a single item from a DynamoDB table using the primary key.

```
aws dynamodb query \
    --table-name <table-name> \
    --key-condition-expression "<partition-key> = :value" \
    --expression-attribute-values '{":value": {"S": "<value>"}}'
```
Retrieves items from a DynamoDB table based on a query using the partition key and optional sort key.

```
aws dynamodb update-item \
    --table-name <table-name> \
    --key '{"<attribute-name>": {"S": "<key-value>"}}' \
    --update-expression "SET <attribute-name> = :value" \
    --expression-attribute-values '{":value": {"S": "<new-value>"}}'
```
Updates an existing item in a DynamoDB table by modifying its attributes.

```
aws dynamodb delete-item \
    --table-name <table-name> \
    --key '{"<attribute-name>": {"S": "<key-value>"}}'
```
Deletes a single item from a DynamoDB table using the primary key.

```
aws dynamodb scan \
    --table-name <table-name>
```
Scans all items in a DynamoDB table and returns results, useful for reading entire datasets.

SNS

```
aws sns list-topics
```
Lists all SNS topics in your AWS account.

```
aws sns create-topic
    --name <topic-name>
```
Creates a new SNS topic with the specified name.

```
aws sns delete-topic
    --topic-arn <topic-arn>
```
Deletes an SNS topic and all its subscriptions.

```
aws sns publish
    --topic-arn <topic-arn>
    --message <message>
```
Publishes a message to a specified SNS topic or directly to an endpoint like an SMS number.

```
aws sns publish --topic-arn arn:aws:sns:us-west-2:123456789012:MySNSTopic --message "Hello from SNS!"
```

```
aws sns subscribe
    --topic-arn <topic-arn>
    --protocol <protocol>
    --notification-endpoint <endpoint: queue-arn>
```
Subscribes an endpoint (e.g., email, SMS, or Lambda) to a topic so that the endpoint receives messages from the topic.

Protocols: email, sms, lambda, http, https

```
aws sns subscribe --topic-arn arn:aws:sns:us-west-2:123456789012:MySNSTopic --protocol email
```

```
    --notification-endpoint example@example.com
```

aws sns unsubscribe

--subscription-arn <subscription-arn>

Removes a subscription from an SNS topic.

```
aws sns unsubscribe --subscription-arn
arn:aws:sns:us-west-2:123456789012:MySNSTopic:2bcfbf39-05c5-41ec-9b59-6788b5fd4c6d
```

aws sns set-subscription-attributes

--subscription-arn <subscription-arn>

--attribute-name <attribute-name>

--attribute-value <attribute-value>

Sets or modifies the attributes of an existing subscription.

aws sns get-topic-attributes

--topic-arn <topic-arn>

Retrieves the attributes of an SNS topic.

aws sns get-subscription-attributes

--subscription-arn <subscription-arn>

Retrieves the attributes of a subscription.

SQS

aws sqs receive-message

--queue-url <queue-url>

--max-number-of-messages 10 Retrieve multiple messages at once (up to 10).

--visibility-timeout 120 Set the visibility timeout for individual messages when retrieving them

Retrieves one or more messages from the specified queue.

This retrieves the message but doesn't delete it from the queue.

SNS Message ID (publishResponse.messageId()):

When you publish a message to an Amazon SNS topic, the message is assigned a Message ID by SNS.

This ID uniquely identifies the message that was published to the topic.

SQS Message ID (retrieved by receive-message):

When the message is delivered to an SQS queue (subscribed to the SNS topic), SQS generates its own Message ID.

This message ID uniquely identifies the message in the SQS queue, and it's separate from the SNS message ID.

The SNS message ID can be retrieved through the Messages.Body.MessageId field of the response body.

Visibility Timeout:

When a message is received using the aws sqs receive-message command,

it becomes temporarily invisible to other consumers for a specified duration, called the visibility timeout.

(default is 30 seconds, but it's configurable).

During this time, no other consumer can receive the same message.

The visibility timeout can be set to a value between 0 seconds and 12 hours (43,200 seconds).

If the visibility timeout is set to 0 seconds, the message becomes immediately visible again.

Message Deletion:

If the message is not deleted (using aws sqs delete-message) before the visibility timeout expires,

the message will become visible again in the queue,

allowing it to be received and processed by another consumer (or the same one if it's still available).

Example Output:

```
{
    "Messages": [
        {
            "MessageId": "f5d9b6c4-6795-4111-b05f-9c85c4c5d593",
            "ReceiptHandle": "AQEBH2...",
            "MD5OfBody": "e59cbbd4d48a3a47d9d2f1c1ec6bff08",
            "Body": "Hello, World!"
        }
```

```
            ]
        }
```

aws sqs set-queue-attributes \

    --queue-url <queue-url>

    --attributes VisibilityTimeout=60

aws sqs get-queue-attributes

    --queue-url <queue-url>

    --attribute-name All

    Retrieves attributes for a specified queue.

    Example Output:

```
        {
            "Attributes": {
                "ApproximateNumberOfMessages": "1",
                "VisibilityTimeout": "30",
                "CreatedTimestamp": "1631197554",
                "QueueArn": "arn:aws:sqs:us-east-1:123456789012:MyQueue",
                "LastModifiedTimestamp": "1631197554",
                "DelaySeconds": "0"
            }
        }
```

aws sqs delete-message

    --queue-url <queue-url>

    --receipt-handle <receipt-handle>

    Deletes the specified message from the queue.

    ReceiptHandle

        The ReceiptHandle is a unique identifier that AWS SQS assigns to each message when it is received from the queue.

        It serves as a temporary token that must be used to delete or manipulate the message.

    Visibility timeout dependent:

        Once the visibility timeout expires and the message becomes visible in the SQS queue again,

        the previous ReceiptHandle becomes invalid.

        If the visibility timeout expires and the message becomes visible again,

        a new ReceiptHandle will be generated when the message is received.

aws sqs change-message-visibility

    --queue-url <queue-url>

    --receipt-handle <receipt-handle>

    --visibility-timeout <timeout>

aws sqs list-queues

    Lists all the SQS queues in your AWS account.

```
        {
            "QueueUrls": [
                "http://sqs.us-east-1.amazonaws.com/123456789012/MyQueue",
                "http://sqs.us-east-1.amazonaws.com/123456789012/MyFIFOQueue.fifo"
            ]
        }
```

aws sqs create-queue

    --queue-name <queue-name>

        Creates a new SQS queue.

    aws sqs create-queue --queue-name <queue-name>.fifo --attributes FifoQueue=true

    For FIFO Queue (must end with .fifo)

    Example Output:

```
        {
            "QueueUrl": "http://sqs.us-east-1.amazonaws.com/123456789012/MyQueue"
        }
```

aws sqs send-message

--queue-url <queue-url>

--message-body "Hello, World!"

Sends a message to the specified queue.

Example Output:

```
{
    "MD5OfMessageBody": "e59cbbd4d48a3a47d9d2f1c1ec6bff08",
    "MessageId": "f5d9b6c4-6795-4111-b05f-9c85c4c5d593"
}
```

aws sqs delete-queue

--queue-url <queue-url>

Deletes the specified queue and all its messages.

aws sqs purge-queue

--queue-url <queue-url>

Deletes all messages in the specified queue.

aws sns list-subscriptions-by-topic

--topic-arn <topic-arn>

List subscriptions associated with a specific topic to see which endpoints are receiving messages.

Others

Git

The Tcl/Tk GUIs (e.g. gitk, git-gui) are now in the `git-gui` formula.
Subversion interoperability (git-svn) is now in the `git-svn` formula.

zsh completions and functions have been installed to:
  /usr/local/share/zsh/site-functions

Groovy

You should set GROOVY_HOME:
  export GROOVY_HOME=/usr/local/opt/groovy/libexec

Node

OpenJDK

For the system Java wrappers to find this JDK, symlink it with
  sudo ln -sfn /usr/local/opt/openjdk@11/libexec/openjdk.jdk /Library/Java/JavaVirtualMachines/openjdk-11.jdk

openjdk@11 is keg-only, which means it was not symlinked into /usr/local,
because this is an alternate version of another formula.

If you need to have openjdk@11 first in your PATH, run:
  echo 'export PATH="/usr/local/opt/openjdk@11/bin:$PATH"' >> ~/.zshrc

For compilers to find openjdk@11 you may need to set:
  export CPPFLAGS="-I/usr/local/opt/openjdk@11/include"

jdk directory: /usr/local/opt/openjdk@8/libexec/openjdk.jdk/Contents/Home
/opt/homebrew/opt/openjdk@17/usr/local/opt/openjdk@8/libexec/openjdk.jdk/Contents/Home
jdk directory: /usr/lib/jvm/java-17-openjdk

NVM

Please note that upstream has asked us to make explicit managing
nvm via Homebrew is unsupported by them and you should check any
problems against the standard nvm install method prior to reporting.

You should create NVM's working directory if it doesn't exist:
  mkdir ~/.nvm

Add the following to your shell profile e.g. ~/.profile or ~/.zshrc:
  export NVM_DIR="$HOME/.nvm"
  [-s "/usr/local/opt/nvm/nvm.sh"] && \. "/usr/local/opt/nvm/nvm.sh" # This loads nvm
  [-s "/usr/local/opt/nvm/etc/bash_completion.d/nvm"] && \. "/usr/local/opt/nvm/etc/bash_completion.d/nvm" # This loads nvm bash_completion

You can set $NVM_DIR to any location, but leaving it unchanged from
/usr/local/Cellar/nvm/0.40.1 will destroy any nvm-installed Node installations
upon upgrade/reinstall.

Type `nvm help` for further information.

JQ

echo '{"key": "value"}' | jp key "value".

K6

k6 run script.js.

Tree

Show dictionary structure

Nodejs	
D:\Server\nodejs	目录必须存在
C:\Users\saidake\.npmrc	修改 npm 配置

NVM Installation

https://github.com/coreybutler/nvm-windows Install nvm windows

根目录下新建两个目录 node-global（npm 全局安装位置） node-cache（npm 缓存路径）

设置环境变量 xxx（安装根目录 node 和 npm） /node-global（全局目录---webpack）

npm 配置全局包目录和缓存目录 npm config set prefix "F:\XXX\node-v8.11.3-win-x64\node-global" 修改全局安装路径（默认值为 windows 的 roaming 可执行目录：C:\\Users\\hz46873\\AppData\\Roaming\\npm）

npm config set cache "F:\XXX\node-v8.11.3-win-x64\node-cache" 修改缓存路径

安装包的时候，npm 按照如下顺序读取这些配置文件

NPM Configuration

loading priority

Global Configuration ($PREFIX/etc/npmrc)

This is the global .npmrc file located in the global installation directory of npm.
It typically resides at /etc/npmrc on Unix systems or at %APPDATA%/npm/etc/npmrc on Windows.
The configuration here applies to all users on the system.
nvm default location: $NVM_DIR/versions/node/<node_version>/etc/npmrc
nvm default prefix: $NVM_DIR/versions/node/<node_version>/lib/node_modules

User Configuration (~/.npmrc)

This is the .npmrc file located in the user's home directory.

For Unix systems, it's typically found at ~/.npmrc, while on Windows, it's located at %USERPROFILE%/.npmrc.

This file is used to store user-specific configurations, such as registry settings or authentication tokens.

Project Configuration (<project root>/.npmrc)

This .npmrc file is located in the root directory of your project.

It applies only to the current project and overrides both global and user-level configurations.

It is often used to set project-specific configurations like custom registries or package access control.

Environment-Specific Configuration (npm_config_* environment variables)

Environment variables prefixed with npm_config_ can override any setting specified in .npmrc files.

For example, setting npm_config_registry would override any registry setting in the .npmrc files.

.npmrc

annotation

registry=https://artifactrepository.citigroup.net/artifactory/api/npm/npm 公共源

https://registry.npm.taobao.org

@test:registry=https://npm.xx.com

@icg-123456:registry=https://a.test.net/artifactory/api/npm/npm-teamdev-local/ 以@icg-123456 开头的包从registry=https://a.test.net 这里下载，其余全去 公共源 下载

//a.test.net/artifactory/api/npm/npm-teamdev-local/:_password=VA2a04NE5EOGR3Y23333vddddVODZHNWMp;

//a.test.net/artifactory/api/npm/npm-teamdev-local/:username=123

//a.test.net/artifactory/api/npm/npm-teamdev-local/:email=123@citi.com

//a.test.net/artifactory/api/npm/npm-teamdev-local/:always-auth=true

//www.a.test.net/artifactory/api/npm/npm-teamdev-local/:_password= VA2a04NE5EOGR3Y23333vddddVODZHNWMp;

Need to add authentiation for different addresses.

//www.a.test.net/artifactory/api/npm/npm-teamdev-local/:username=123

//www.a.test.net/artifactory/api/npm/npm-teamdev-local/:email=123@citi.com

//www.a.test.net/artifactory/api/npm/npm-teamdev-local/:always-auth=true

prefix=F:\XXX\node-v8.11.3-win-x64\node-global Custom npm global directory

prefix=~/.nvm/versions/node/$(nvm version) NVM setup

export PATH=~/.npm-global/bin:$PATH Linux Environment Variable

cache=F:\XXX\node-v8.11.3-win-x64\node-cache Custom npm cache directory

cache=~/.npm-cache NVM setup

proxy=http://webproxy.test.net:8080/ Configure proxy to forward your download request to another address.

http-proxy=http://webproxy.test.net:8080/ 配置代理

disturl=https://www.artifact.net/xxx

sass_binary_site=https://www.artifact.net/xxx

strict-ssl=false

globalconfig=C:\Program Files\nodejs\etc\npmrc 全局配置，导入外部配置.npmrc

auto-install-peers=true 在 npm 3 中，不会再强制安装 peerDependencies （对等依赖）中所指定的包，而是通过警告的方式来提示我们

nrm

npm install nrm -g　源切换工具

nrm current　查看当前源

NPM Version

Base　此版本表示该软件仅仅是一个假页面链接，通常包括所有的功能和页面布局，但是 页面中的功能都没有做完整的实现，只是做为整体网站的一个基础架构。

Alpha　软件的初级版本，表示该软件在此阶段以实现软件功能为主，通常只在软件开发者 内部交流，一般而言，该版本软件的 Bug 较多，需要继续修改，是测试版本。测试 人员提交 Bug 经开发人员修改确认之后，发布到测试网址让测试人员测试，此时可 将软件版本标注为 alpha 版。

Beta　该版本相对于 Alpha 版已经有了很大的进步，消除了严重错误，但还需要经过多次 测试来进一步消除，此版本主要的修改对象是软件的 UI。修改的的 Bug 经测试人 员测试确认后可发布到外网上，此时可将软件版本标注为 beta 版。

RC　该版本已经相当成熟，基本上不存在导致错误的 Bug，与即将发行的正式版本相差无几。

Release　该版本意味“最终版本”，在前面版本的一系列测试版之后，终归会有一个正式的版本，是最终交付用户使用的一个版本。该版本有时也称标准版。

"boo": "2.3.1",　必须匹配这个版本（beta 版本互相依赖时，使用 overrides 覆盖多个依赖版本）

"boo": "~2.3.1",　安装 2.3.1 以上的版本，但是不安装 2.4.0

"boo": "^2.3.1",　安装 2.3.1 以上的版本，但是不安装 3.0.0

"foo": "1.0.0 - 2.9999.9999",　指明版本范围

"bar": ">=1.0.2 <2.1.2",　必须大于等于 1.0.2 版本且小于 2.1.2 版本

"baz": ">1.0.2 <=2.3.4",　必须大于 1.0.2 版本且小于等于 2.3.4 版本

"thr": "2.3.x",

"qux": "<1.0.0 || >=2.3.1 <2.4.5 || >=2.5.2 <3.0.0",

"asd": "http://asdf.com/asdf.tar.gz",　在版本上指定一个压缩包的 url，当执行 npm install 时这个压缩包会被下载并安装到本地。

"til": "~1.2",

"elf": "~1.2.3",

"two": "2.x",

"lat": "latest",　安装最新版本

"dyl": "file:../dyl",　使用本地路径

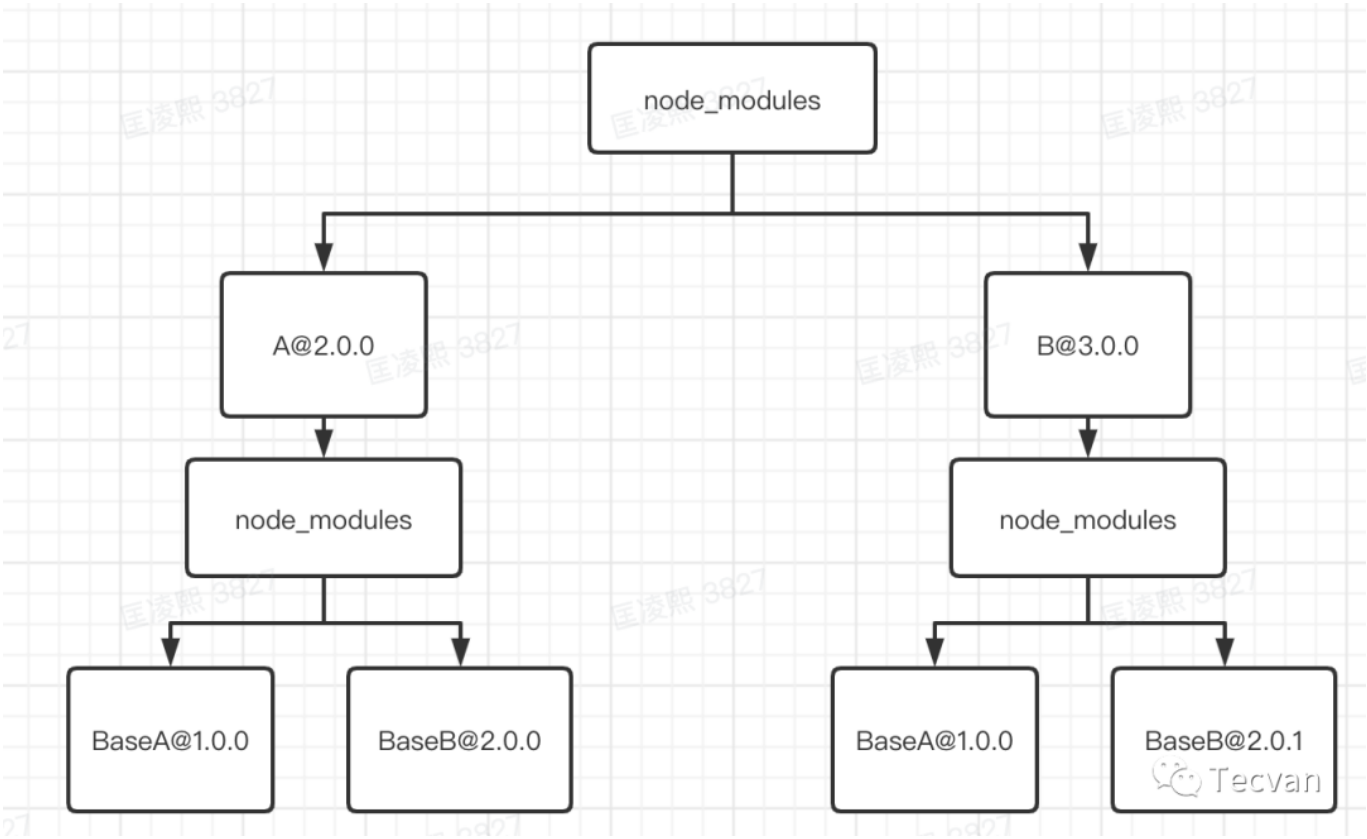
"adf": "git://github.com/user/project.git#commit-ish"　使用 git URL 加 commit-ish

NPM Principle

npm2 无论 A 和 B 的子依赖是否是同一版本，都会分别直接生成树结构：

A@2.0.0：BaseA@1.0.0 + BaseB@2.0.0

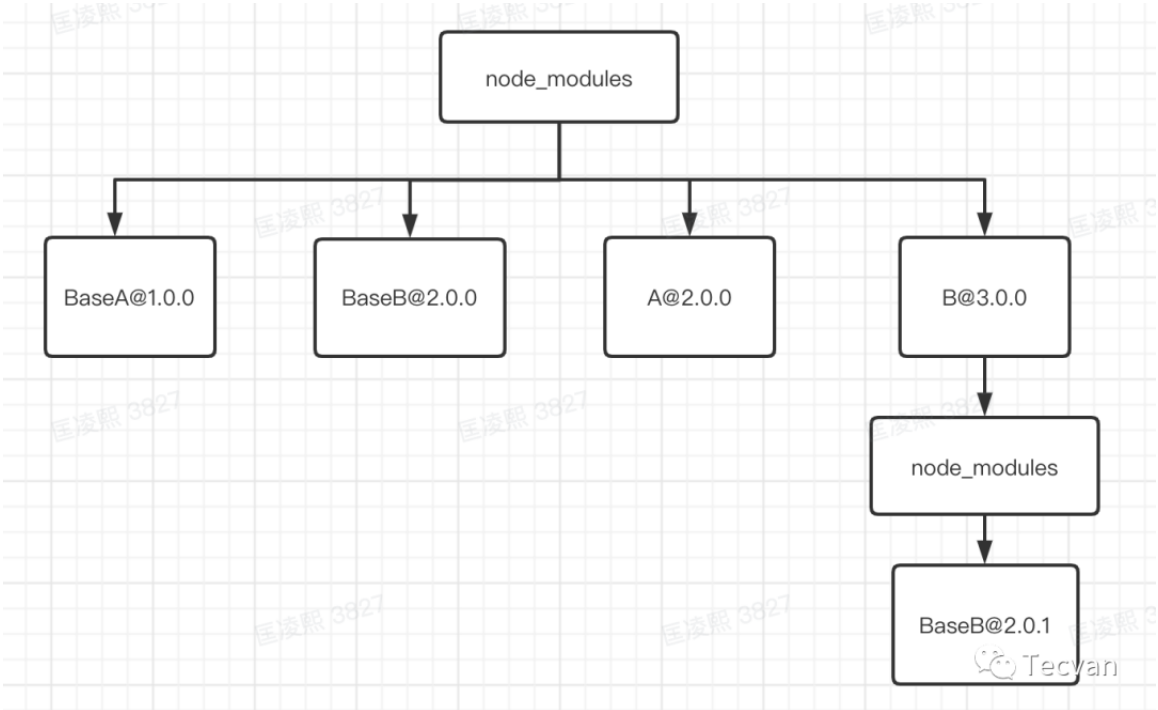
B@3.0.0：BaseA@1.0.0 + BaseB@2.0.1



npm3 将树打平，将依赖扁平化（公共依赖提取到外部）：

问题：项目原本的 node_modules 结构不够直观

依赖不安全，我们可以使用依赖文件中并没有声明的 npm 包

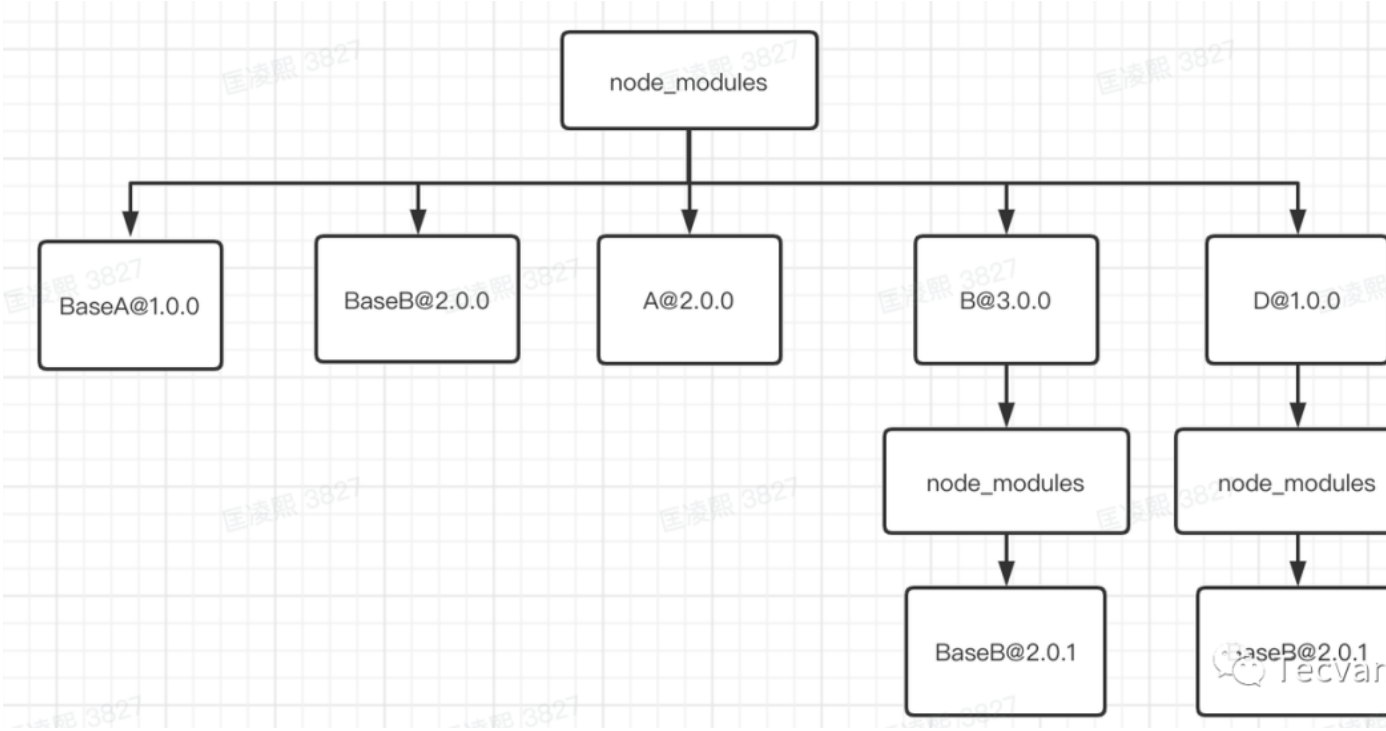


同时引用了同一个包的多个不同版本，会按照字典排序优先提取 BaseB@2.0.0，

A@1.0.0：BaseA@1.0.0 BaseB@2.0.0

B@1.0.0：BaseA@1.0.0 BaseB@2.0.1

D@1.0.0：BaseA@1.0.0 BaseB@2.0.1



Commands

npm

npm –v 查看版本

npm init [-y] 初始化向导生成 package.json 包说明文件（创建 webpack.config.js）【-y 跳过向导】

npm install aa bb 下载安装 package.json 里的所有依赖项到当前项目目录下载包 **node_modules**（安装中止时会有受损的包，全局包安装位置 node-11.13.2\node-global\node_modules）

npm install @11ty/eleventy@latest # 最新版本

npm install @11ty/eleventy@beta # 最新 beta 版本

aa@3.3.3 指定版本

-g 全局包

--save-prod 在 package.json 中添加依赖项

--save-dev 本地包--不打包上线

--save-exact 的作用就是固定依赖包的版本

--legacy-peer-deps 在 v7 中引入的，目的是绕过 peerDependency 自动安装，忽略项目中引入的各个 modules 之间的相同 modules 但不同版本的问题并继续安装，保证各个引入的依赖之间对自身所使用的不同版本 modules 共存

--force Overwrite a module if it already exists

npm uninstall [--s -g] xxx 删除包 【--s 同时删除此包的所有依赖项 -g 包括全局的包】

npm update [-s] xxx 更新一个包

npm run [script] 快捷执行 script 中的命令 【npm run ./node_modules/.bin/eslint 执行本地包的命令】

npm list 查看安装包列表

-g Show global packages

-a Show nested dependency packages

npm config list -l 查看配置（npm config set registry https://registry.npm.taobao.org 设为淘宝服务器下载）

npm config get xxx 获取属性【 prefix .npmrc 全局配置文件路径】

npm config set xxx　　设置属性【 prefix .npmrc 全局配置文件路径】
registry　https://www.xxxxxx.com　　@:registry http://www.xxxxxx.com
npm login　　登录配置的 registry

npm link
This creates a symbolic link to your package in the global node_modules, allowing you to run your CLI command from anywhere in the terminal.
linux：/usr/local/lib/node_modules/
npm unlink　xxx　　当前目录下解除模块链接（package.json 内的项目名）

init --scope=@my-org　　初始化机构
npm publish --access public　　发布第三方包（--access 机构发布）
npm install git+https://github.com/sunxiaochuan/koatest.git　　通过 npm 下载 git 包
npm view lef **version**　　打印 npm 仓库中包版本

npx wbpack -v　　直接执行 node-modules 内的包（目录执行：node-modules/.bin/webpack -v　）
tyarn　cnpm　　都为淘宝源
https://registry.npmjs.org/-/package/create-react-app/dist-tags　　获取 npm 包版本信息
TIPS: @myorg/mypackage　　组织名/包名（@发布，不用担心与其他包名冲突）
修改项目名：　修改 package.json　name　　npm install
npm 无法 login　删除~/.npmrc

nvm

nvm ls-remote　　This will list all the versions of Node.js that are available for installation.
nvm list [available]　　List the node.js installations.
nvm version　　Displays the current running version of NVM for Windows.
nvm current　　Display active version.
nvm install <version> [arch]　　The version can be a specific version, "latest" for the latest current version, or "lts" for the most recent LTS version.
Optionally specify whether to install the 32 or 64 bit version
Add --insecure to the end of this command to bypass SSL validation of the remote download server.
--arch　The CPU architecture (defaults to system arch). (x86 for 32-bit, x64 for 64-bit). Set [arch] to "all" to install 32 AND 64 bit versions.

```
nvm install 14.17.6 --arch=x64
```

nvm npm_mirror <npm_mirror_url>　　Set the npm mirror.People in China can use
// nvm npm_mirror https://registry.npm.taobao.org
nvm root <path>　　Set the directory where nvm should store different versions of node.js.
Change root directory for version of nodes (without spaces)

node

node -v　　查看版本（安装 node 后自动安装 npm，node 服务器的访问域名默认为本机）
node xxx.js　　运行 js 文件开启服务器

pnpm

项目的 node_modules 文件夹只有当前 package.json 中所声明的各个依赖（的软连接），而真正的模块文件，存在于 node_modules/.pnpm（硬链接）
避免可以访问非法 npm 包

pnpm i 安装依赖
pnpm add xxx 本地安装指定包【-D 本地安装 -g 全局安装】
pnpm remove xxx 移除包【--global 全局包】
pnpm upgrade xxx 更新包【--global 全局包】

ERROR：pnpm 缺少 module，但是 npm install 正常（Error: Cannot find module 'mini-css-extract-plugin'）

yarn

yarn --version
Displays the installed Yarn version.
yarn init
Initializes a project and generates a package.json file. (yarn.lock locks dependency versions)
yarn add <package>
Installs a third-party package and adds it to package.json.
-D Add it as a devDependency.
yarn global add <package>
Installs a package globally.
yarn remove <package>
Removes a package from the project.
yarn global remove <package>
Removes a globally installed package.
yarn upgrade <package>
Updates a specific package to its latest version.
yarn run <script>
Executes a script defined in the scripts section of package.json.
yarn config list
Displays the current configuration settings for Yarn
(e.g., to set the registry: yarn config set registry http://registry.npm.taobao.org to use the Taobao mirror for faster downloads).
yarn create @umijs/umi-app mypro
Equivalent to yarn global add @umijs/create-umi-app and then running @umijs/create-umi-app mypro to create a UmiJS app.
yarn create react-app mypro
Equivalent to yarn global add create-react-app and then running create-react-app mypro to create a React app.

nrm

nrm set-auth [options] <registry> [value]： 使用 base64 编码的字符串或用户名和密码设置自定义镜像源的授权信息。
nrm set-email <registry> <value>： 设置自定义镜像源的邮箱。

nrm add xxx http://baidu.com 添加源
nrm use xxx 使用源
nrm del xxx 删除源
nrm ls 源列表

nginx

Core

Proxy

Forward Proxy (Normal Proxy)

A forward proxy server sits between a client (like a web browser) and the internet.
It acts on behalf of the client to send requests to other servers on the internet.
Usage Scenarios
- It is often used to hide the client's IP address, bypass content restrictions, or cache data for the client.
- The client is aware that it is using a proxy server.

Reverse Proxy
A reverse proxy server sits between the internet and the web server.
It handles incoming requests from clients on behalf of the web server and forwards them to the appropriate backend server.
The client usually does not know it is interacting with a reverse proxy.
Usage Scenarios
- Load Balancing:
  - A reverse proxy can distribute incoming client requests across multiple servers, balancing the load and ensuring no single server is overwhelmed.
- Security:
  - By acting as a shield, a reverse proxy can hide the IP addresses of the backend servers, making them less vulnerable to attacks.
  - It can also provide additional security features like SSL termination, where the reverse proxy handles SSL encryption and decryption.
- Caching:
  - A reverse proxy can cache content from the web server, reducing the load on the backend servers and speeding up response times for clients.
- Compression:
  - It can compress outgoing data to reduce bandwidth usage and improve loading times for clients.
- SSL Termination:
  - The reverse proxy can manage SSL certificates and offload the encryption/decryption process from the backend servers.
- Traffic Filtering:
  - A reverse proxy can filter and route traffic based on various criteria, such as URL paths, request headers, or source IP addresses.

Configuration

nginx.conf

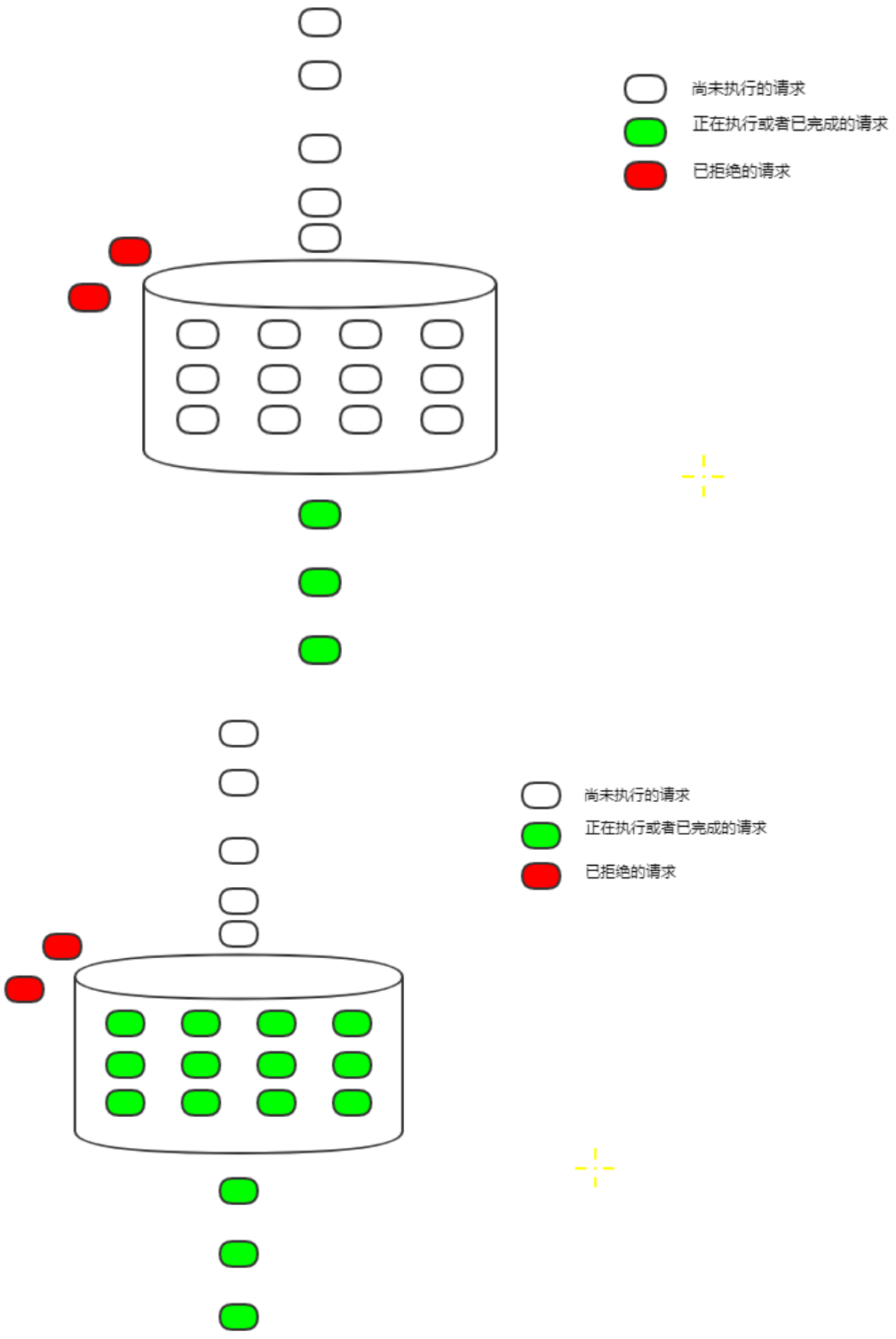
Implement caching directives in Nginx to store the dynamic content in memory or on disk for faster subsequent responses.

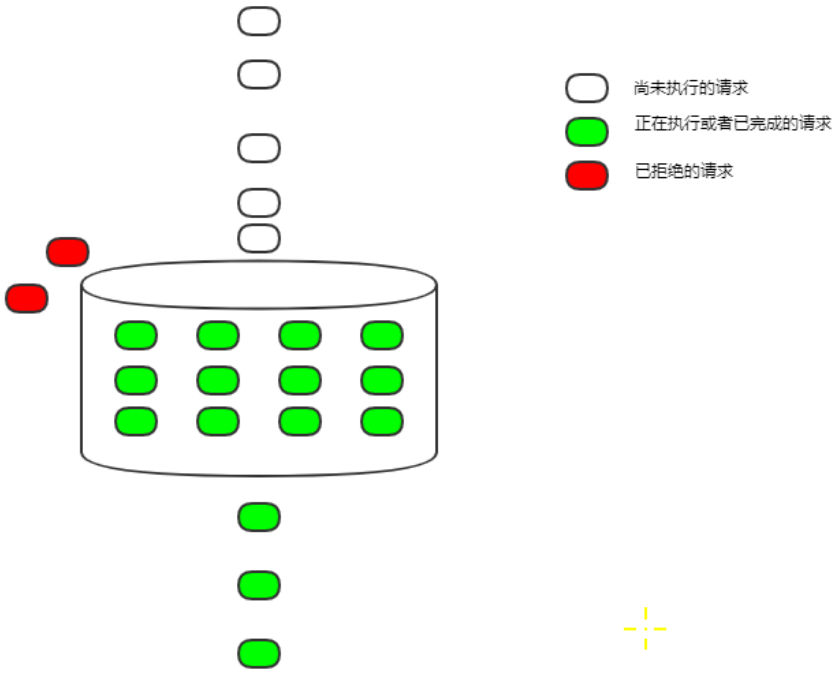
```
proxy_cache_path /path/to/cache levels=1:2 keys_zone=my_cache:10m max_size=10g inactive=60m use_temp_path=off;

server {
    location / {
        proxy_pass http://backend-server:8080;
            # Forwards requests to the backend server listening on http://backend-server:8080.
        proxy_cache my_cache;
            # proxy_cache my_cache;: Enables caching using the my_cache zone defined earlier.
        proxy_cache_valid 200 1h;
            # Sets the caching validity period for successful (200) responses to 1 hour (1h).
            # This means responses with a 200 OK status from the backend will be cached for up to 1 hour.
        proxy_cache_use_stale error timeout updating http_500 http_502 http_503 http_504;
        proxy_cache_bypass $http_cache_control;
        proxy_no_cache $http_pragma $http_authorization;
```

```
    }
}


#全局：配置影响 nginx 全局的指令。

user nobody;                        运行用户，必须和启动 nginx 线程的用户一致（没有权限会导致 403）
worker_processes 2;                 启动的进程数，默认为 1（通常设置成和 cpu 数量相等）

error_log log/error.log debug;          指定日志路径，级别。【这个设置可以放入全局块，http 块，server 块，级别依次为：debug|info|notice|warn|error|crit|alert|emerg 】
pid /nginx/pid/nginx.pid;               指定 nginx 进程运行文件存放地址
events {                #配置影响 nginx 服务器或与用户的网络连接
    accept_mutex on;                    设置网路连接序列化，防止惊群现象发生，默认为 on
    multi_accept on;                    设置一个进程是否同时接受多个网络连接，默认为 off
    use epoll;                          事件驱动模型，select|poll|kqueue|epoll|resig|/dev/poll|eventport    （epoll 是多路复用 IO(I/O Multiplexing)中的一种方式，，仅用于 linux2.6 以上内核,可以大大提高 nginx 的性能）
    worker_connections  1024;       单个后台 worker process 进程的最大并发链接数，默认为 512
}
http{                   #可以嵌套多个 server，配置代理，缓存，日志定义等绝大多数功能和第三方模块的配置。
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=10r/s;             # 定义限流规则 mylimit（漏桶算法，需要在 server 配置内定义漏桶大小，否则的话除了漏出的那个请求，其他请求都会被拒绝）
                $binary_remote_addr         针对客户端 ip 限流；
                zone=ip_limit:10m           限流规则名称为 ip_limit，允许使用 10MB 的内存空间来记录 ip 对应的限流状态；
                rate=10r/s                  限流速度为每秒 10 次请求
                location /login/            对指定路径进行限流


    include         mime.types;                 设定 mime 类型,类型由 mime.type 文件定义
    default_type    application/octet-stream;       默认文件类型，默认为 text/plain


    error_page 404 https://www.baidu.com;       #错误页
    access_log    logs/access.log    main;          服务日志
    access_log log/access.log myFormat;         #输出日志目录
    error_log log/access.log myFormat;          #错误日志目录
    log_format myFormat '$remote_addr–$remote_user [$time_local] $request $status $body_bytes_sent $http_referer $http_user_agent $http_x_forwarded_for'; #自定义日志格式
                $remote_addr  与$http_x_forwarded_for  用以记录客户端的 ip 地址；
                $remote_user                            用来记录客户端用户名称；
                $time_local                             用来记录访问时间与时区；
                $request                                用来记录请求的 url 与 http 协议；
                $status                                 用来记录请求状态；成功是 200，
                $body_bytes_s ent                       记录发送给客户端文件主体内容大小；
                $http_referer                           用来记录从那个页面链接访问过来的；
                $http_user_agent                        记录客户端浏览器的相关信息；
    gzip on;                                    #开启 gzip 压缩
```

```
    client_header_buffer_size       128k;           #设定请求缓冲
    large_client_header_buffers     4 128k;

    sendfile on;                                    允许 sendfile 方式传输文件，默认为 off，可以在 http 块，server 块，location 块。
    sendfile_max_chunk 100k;                        每个进程每次调用传输数量不能大于设定的值，默认为 0，即不设上限。
    keepalive_timeout 65;                           连接超时时间，默认为 75s，可以在 http，server，location 块。
    upstream mysvr {                                调度算法   轮询：每个请求按时间顺序逐一分配到不同的后端服务器，如果后端某台服务器宕机，故障系统被自动剔除，使用户访问不受影响）
      server 127.0.0.1:7878;
      server 192.168.10.121:3333 backup;   #热备
    }
    upstream webhost {                              调度算法  权重：可以根据机器配置定义权重.权重越高被分配到的几率越大
        server 192.168.0.5:6666 weight=2;
        server 192.168.0.7:6666 weight=3;
    }
    upstream webhost {          调度算法 ip_hash：   每个请求按访问 IP 的 hash 结果分配,这样来自同一个 IP 的访客固定访问一个后端服务器,有效解决了动态网页存在的 session 共享问题
        ip_hash;
        server 192.168.0.5:6666 ;
        server 192.168.0.7:6666 ;
    }

    upstream webhost {          调度算法  url_hash(需安装第三方插件) ：此方法按访问 url 的 hash 结果来分配请求，使每个 url 定向到同一个后端服务器，可以进一步提高后端缓存服务器的效率.Nginx 本身是不支持 url_hash 的,如果需要使用这种调度算法，必须安装 Nginx  的 hash 软件包

        server 192.168.0.5:6666 ;
        server 192.168.0.7:6666 ;
        hash $request_uri;
    }
    upstream webhost{    调度算法 fair(需安装第三方插件)：这是比上面两个更加智能的负载均衡算法。此种算法可以依据页面大小和加载时间长短智能地进行负载均衡,也就是根据后端服务器的响应时间来分配请求，响应时间短的优先分配。Nginx 本身是不支持 fair 的，如果需要使用这种调度算法，必须下载 Nginx 的 upstream_fair 模块
    }
    server {
        listen 443 ssl http2 default_server;
        listen          [::]:443 ssl http2 default_server;
        server_name sdk.vin www.jhxblog.cn;                 #证书绑定的域名。

        ssl_certificate "/etc/pki/nginx/sdk.vin_bundle.pem";          #证书
        ssl_certificate_key "/etc/pki/nginx/sdk.vin.key";       #证书

        ssl_session_timeout 5m;
        ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
        ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;          #表示使用的 TLS 协议的类型。
        ssl_prefer_server_ciphers on;
        ssl_session_cache shared:SSL:1m;
```

```
        root html;
        index index.html index.htm;
        location / {
          add_header 'Access-Control-Allow-Origin' *;
          add_header 'Access-Control-Allow-Credentials' 'true';      #允许带上 cookie 请求
          add_header 'Access-Control-Allow-Methods' *;               #允许请求的方法，比如 GET/POST/PUT/DELETE
          add_header 'Access-Control-Allow-Headers' *;               #允许请求的 header
            root html;   #Web 网站程序存放目录。
            index index.html index.htm;
        }
    }

    server {                        #配置虚拟主机的相关参数，可多个。
        listen          80;                                  #监听端口
        server_name   sdk.vin;                              #监听地址，通过请求头中的 HOST 字段，匹配到特定的 server 块，转发到对应的应用服务器中去   // *.test.com
        client_max_body_size 1024M;
        root /kong;        #静态文件，location 的 root 生效时，此配置不生效（自动去/kong 目录下找文件）

        include /etc/nginx/default.d/*.conf;
        location /xxx {              #匹配请求路由（）
            root    /kong;             #静态文件目录，覆盖 server 的 root 配置（自动去/kong/xxx 目录下找文件）
            alias    /kong;             #别名静态文件目录（自动去/kong 目录下找文件）
            index   index.html index.htm;                        #默认首页
            rewrite      ^/name/(.*)$ /user/$1 break;             #匹配正则重写，重写用户请求整个路径（可以先 break 重写再转发，注意防止转发后再次重复匹配）【last   跳出 location ，重新开始再走一次刚刚的行为   break   不会跳出 location    redirect   临时重定向   permanent   永久性重定向】
            proxy_pass   http://127.0.0.1:8080;                  #匹配路径转发，重写主机（加上了/，则会把匹配的路径部分也给代理走          没有/，相当于是绝对根路径，则 nginx 不会把 location 中匹配的路径部分代理走）
                                                                        /xxx      http://127.0.0.1:8080
                                                                    http://sdk.vin/xxx/abc.html   ==>
                                                                    http://127.0.0.1:8080/xxx/abc.html    （不加/   只会替换原本的 host）
                                                                        /xxx/     http://127.0.0.1:8080/
                                                                    http://sdk.vin/xxx/abc.html   ==>
                                                                    http://127.0.0.1:8080/abc.html          （加上/   会替换请求路径）

            deny 127.0.0.1;                                     #拒绝的 ip
            allow 172.18.5.54;                                   #允许的 ip
            limit_req zone=ip_limit burst=12 nodelay;        指定限流规则
                    burst        桶队列，超出队列的请求会被拒绝（没有 nodelay 的话，如果有 10 次请求同时到达，它们会依次排队执行，每 100ms 执行 1 个）
                    nodelay    只要入桶就开始执行，并发执行数为 burst（限流不均匀。如果有 12 个请求同时到达，那么这 12 个请求都能够立刻执行，然后后面的请求只能匀速进桶）
                    delay=4    从桶内第 5 个请求开始 delay，并发执行数为 4（调整允许并发执行的请求的数量）
```


尚未执行的请求
正在执行或者已完成的请求
已拒绝的请求
尚未执行的请求
正在执行或者已完成的请求
已拒绝的请求




```
        }
        location ~^/(doc.html|swagger-resources|v2/api-docs)$ {
                        client_max_body_size 100m;
                        proxy_pass http://127.0.0.1:82;
                        proxy_redirect default ;
                        proxy_http_version 1.1;
                        proxy_set_header Upgrade $http_upgrade;
                        proxy_set_header Connection 'upgrade';
                        proxy_set_header Host $host;
                        fastcgi_hide_header Access-Control-Allow-Origin;
                        proxy_hide_header Access-Control-Allow-Origin;
                        add_header 'Access-Control-Allow-Origin' $http_origin always;
                        proxy_set_header X-Real-IP $remote_addr;
                        proxy_set_header X-Forwarded-Proto $scheme;
                        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            client_max_body_size 100m;
            proxy_pass      http://127.0.0.1:11220/;
            proxy_redirect default ;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_cache_bypass $http_upgrade;
        }
    }
}
```

Command

默认日志目录：/var/log/nginx

nginx -h 打印帮助信息

nginx -c pathxxx 使用替代的配置文件而不是默认文件

nginx -e pathxxx 使用替代的错误日志文件储存日志，而不是默认文件 // nginx -e stderr 选择标准错误日志文件

nginx -g 设置全局配置命令 // nginx -g "pid /var/run/nginx.pid; worker_processes `sysctl -n hw.ncpu`;"

nginx prefix pathxxx 设置 nginx 路径前缀，即保存服务器文件的目录（默认为/usr/local/nginx）

nginx -q 在配置测试期间 压制非错误消息

nginx -s 发送一个信号给主线程

- stop 迅速停止
- quit 优雅停止
- reload 停止老进程，使用新配置文件开始新进程
- reopen 重新打开日志文件

nginx -t 测试配置文件，检查配置文件的语法是否正确，然后尝试打开配置中引用的文件。

nginx -T 与-t 相同，但另外将配置文件转储到标注输出

nginx -v 打印 nginx 版本

nginx -V 打印 nginx 版本，编译器版本，配置参数

Installation

vim /etc/yum.repos.d/nginx.repo

```
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true

[nginx-mainline]
name=nginx mainline repo
baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
gpgcheck=1
enabled=0
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
```

```
yum -y install yum-utils
yum-config-manager --enable nginx-mainline
yum -y install nginx
```

git

Configuration

Secret Key

ssh-keygen -t rsa -C "saidake@qq.com" 生成私钥 id_rsa 和公钥 id_rsa.pub， 再在 git 网页添加 SSH（ssh-keygen.exe 和 cat.exe 在 git 的 usr/bin 目录 ）【known_hosts 内的地址自动用本机公钥认证，认证失败提示输入密码】

-t xxtype 指定生成密钥类型 rsa1(SSH-1)，rsa(SSH-2)，dsa(SSH-2)

-f xxx 指定密钥文件名

-b 233 指定密钥长度。对于 RSA 密钥，最小要求 768 位，默认是 2048 位，DSA 密钥必须恰好是 1024 位(FIPS 186-2 标准的要求)。

-C xxx 提供一个新注释

-N "xxx" 提供一个新的密语

ssh -T git@github.com test git connection

ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.6.221 Write the key to the "~/.ssh/authorized_keys" file on the remote machine to establish an SSH connection.

-i ~/.ssh/id_rsa.pub 指定公钥

-p 8099 远程端口

ssh-agent bash 使用 ssh-agent 命令行

ssh-add ~/.ssh/kk 把专用密钥添加到 ssh-agent 的高速缓存中(linux 需要，**git://** 协议 clone 获得的文件 push 时会 remote error **git@** 协议 push 需要 SSH Key 来验证)

-l 查看加入 ssh-agent 的密钥列表

-L 查看加入 ssh-agent 的公钥列表

-T git@github.com 验证认证是否成功

ERROR: WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! 删除对应缓存 ip 的密钥

Global configuration

git config - core.longpaths true

--global Applies to all repositories

--unset Remove the current configuration entirely

core.autocrlf false Avoid automatic conversion of newline character LF to CRLF

core.autocrlf input 提交时转换为 LF，检出时不转换

core.autocrlf false 提交检出均不转换

// git rm –cached -r . git reset –hard make your changes take effect

user.name "saidake" Configure username

user.email "saidake@qq.com" Configure email

credential.helper store

gc.reflogExpire never reflog 用不过期

pull.rebase false

false

This sets the default behavior to merge when you run git pull.

true

This sets the default behavior to rebase when you run git pull.

pull.ff only

This ensures that git pull uses a fast-forward merge only.

~/.ssh/config

Host github.com 服务器地址为 github 地址
Hostname ssh.github.com 服务器地址为 github 地址
User saidake
AddKeysToAgent yes
port 443 修改端口为 443
IdentityFile ~/.ssh/id_rsa 公匙文件路径
PreferredAuthentications publickey 采用公匙

.gitignore

Priority Order (Highest to Lowest)

Project .gitignore (Current directory)

.gitignore files in subdirectories refine or override specific rules for their directory but do not completely replace the parent .gitignore.

Parent directory .gitignore

.git/info/exclude (Local ignore rules for a single repository)

Global ignore (core.excludesfile) (User-wide ignore settings)

Apply Changes Immediately

```
git rm -r --cached .
git add .
```

Ignore Rules

*.log	Ignores all .log files in any directory of the project..
debug.log	Ignores all files named debug.log in any directory of the project.
node_modules/	Ignores all node_modules directories in any directory of the project..
build/	Ignores all build directories in any directory of the project..
logs/	Ignores any logs/ directory, regardless of its location.
*.tmp	Ignores all .tmp files in any directory.
*.swp	Ignores Vim swap files.
/config.json	Ignores config.json in the root but not in subdirectories.
/bin/	Ignores only the bin/ directory in the root.
logs	Ignores both logs files and directories in any directory of the project.
logs/	Ignores only the logs directory in any directory of the project.
logs/*	Ignores all files inside logs/, but keeps the directory itself.
!important.log	Forces important.log to be included even if *.log is ignored.
!docs/README.md	Includes README.md in the docs/ folder while ignoring other files in docs/.
~$*	Ignores temporary files starting with ~$ (common in Office files).

Command

Git Structure

工作区（项目目录） **暂存区**（存储在 index.lock 定位的 objects 目录）
本地仓库 远程仓库

Initial Configuration

git lfs install

git init adds git management tool for the current directory（All fiels of the current directory will be placed int the management and tracking of git management tool, The first submission requires git init）

If you copy the .git folder of another project to the current project, the current project will not to

be able to be submmited.

git status 查询所有文件的状态（是否含未跟踪文件）

git config --list 查看 git 配置

git config <key> <value> Configure a key-value pair

git config --global <key> <value> Configure a global key-value pair

--global --list View global configuration

--get core.autocrlf Get field value

git rm **-r --cached .** 删除本地缓存文件（--cached 不会删除文件，只会让文件脱离 git 管理，修改.gitignore 后需要删除缓存）

git remote add <origin> git@github.com:saidake/aaa.git 添加一个远程仓库【git://github.com/saidake/aaa.git 或 git@github.com:saidake/aaa.git】

git remote rm <origin> 删除一个远程仓库

git remote –v 查询所有远程仓库【-v 显示详细地址 fetch 和 push】

git remote rename <origin> <new-origin> 修改仓库名

git remote show origin 显示仓库信息（默认 pull 的分支和 push 的分支）

git remote set-url origin <new-url>

Branch Operation

git clone git@github.com:saidake/aaa.git [xxpath] 克隆 远程仓库文件夹 到当前目录（生成 xxx 目录，本地拉取得文件含有所有分支，此时仓库名默认为 origin，搭配 git checkout 命令使用） 【-b testbranch 指定分支】

git checkout <branch> switch to a branch（The modified files must to be commited before checkout）

-b <branch> <remote>/<branch> If the new branch does not exist, create a new branch, pull the code from the remote origin to the new branch,

and swi0tch to the new branch. (pull the remote branch locally in advance to avoid the checking commit error)

// git checkout -b feature/test origin/release/remote-test

// git checkout -b feature/test Copy current branch to another branch and then checkout to another branch.

-B <branch> <remote>/<branch> If the new branch does not exist, create a new branch, otherwise, it is reset.

--track / -t <remote>/<branch> create a new local branch associated with a remote branch.

(pull the remote branch locally to avoid the checking commit error)

(branch name cannot already exist)

git checkout --ours {codefiles} 保留当前分支代码，适用于 merge 和 stash 冲突 . 代表全部文件 (保留 develop 代码)

git checkout --theirs {codefiles} 保留要合并分支代码（保留 feature 的代码）

git branch Query the current lcoal branches.

<branch-name> Create a local branch（Copy the content of the current branch to a new branch by default.）

-list <pattern> The pattern will be used as a shell wildcard to restrict the output to matching branches. If multiple patterns are given, a branch is shown if it matches any of the patterns.

-r Query remote branch

-vv When in list mode, show sha1 and commit subject line for each head, along with relationship to upstream branch(if any).

-a Query all local and remote branches.

-d <branchname> Delete a branch, The branch must be fully merged in its upstream branch, or in HEAD if not upstream wa set.

// git branch -d -r origin/todo origin/html origin/man Delete the remote-tracking branches. The next fetch or pull will create them again unless you configure them not to. See git fetch.

-D <branchname> Delete an unneeded branch.

// git branch -D test Delete the “test” branch even if the “master” branch.

-m <branchname> <newbranchname> Modify branch name.

--set-upstream-to=origin/next <branchname> -u origin/next <branchname> Set the remote tracking branch for the specified branch, If no <branchname> is specified, then it defaults to the current branch.

// git branch | grep feature* | xargs git branch -D Delete all matching branches.

git merge origin/develop

Git tries to **automatically merge** changes from other branch into your current branch.

```
git merge dev (executeed in dev-old branch)
```

All changes in the dev branch (that are not already in dev-old) are **merged** into dev-old.

These changes will now appear as **new changes** in your dev-old branch, even if they are not related to your own modifications.

As a result, when you create a pull request from dev-old to dev, all the changes from dev that were merged into dev-old (but are already in dev) are shown as **modified files**, alongside your changes.

--allow-unrelated-histories Allow merging different branches from different repositories.

--abort

git reset --hard a2177a669cbf7d5e2284700b3fa5176dc6be060b 将 HEAD 移动到此次提交【--soft 保留文件修改】

git stash [save "msg"] 暂时存储当前分支所有修改，当前项目的所有修改会立即消失（git add 之前 save 添加暂存注释）

git stash list 查看 stash 列表

git stash pop 2 应用修改到当前分支，并删除这个 stash，默认为最新的一次 stash（默认为第一个 stash@{0}，可以在此之前 git pull 拉取再恢复之前的修改）

git stash apply 0 应用修改到当前分支，但不会把存储从存储列表中删除

git stash drop 1 删除 stash@{1}存储

git stash clear 删除所有缓存的 stash

git stash show -p stash@{3}

Commit Operation

git push origin HEAD --force 修改远程 HEAD 指向为当前 commit

git log 查看提交历史记录

--pretty=oneline 以精简模式显示

--abbrev-commit 仅显示 SHA-1 的前几个字符，而非所有的 40 个字符

--graph 显示 ASCII 图形表示的分支合并历史

--follow -p com/dd/test.txt 查看每次详细修改内容的 diff

--stat 查看提交统计信息

-p -2 查看最近两次详细修改内容的 diff

git reflog 查看记录

--date=iso 显示时间

git blame com/dd/test.txt 可以显示文件的每一行最后修改的版本和作者。

git diff <compare-branch> <source-branch> [<file-path>]

Compare the differences between two branches. (exclude local files in staging area)

--name-only Only display name.

// git diff develop master --name-only

// git diff develop master > patch.txt Create a patch file that contains all of the changes between the two branches.

// git apply patch.txt Apply changes.

git reset --hard 2b504bee 恢复之前一次操，不带参数为清空缓存区，恢复最新的一次 commit

revert 命令意思是撤销某次提交，它会产生一个新的提交，虽然代码回退了，但是版本依然是向前的

// git revert HEAD 撤销最近一次提交

// git revert HEAD~1 撤销上上次的提交，注意：数字从 0 开始

git reabase <upstream-branch> [<current-branch>] stash your modifications, reset your branch to upstream-branch and merge each of your commits.

--continue 解决合并冲突，必须先 git add.

--abort 会放弃合并，回到 rebase 操作之前的状态，之前的提交的不会丢弃 合并冲突，需要提前执行 git add .

--onto other 创建新的 commit 起点，没有指定默认为上行分支 upstream

-i 进入交互模式（:cq 带一个错误码退出 ）

// git rebase -i HEAD~6 合并 6 个分支

//

pick 保留分支（越向上的分支越是新提交的）

s, squash 和上方 pick 分支合并

r, reword 使用该条 commit，但是修改提交信息

e, edit 使用该条 commit，但是过程中会停下来修改一些属性

f, fixup 类似 squash，但是会丢弃该条提交的记录信息

pick 保留分支

git add

Add files to the staging area and track files (only tracked files can detect conflicts).

. Adds all new and modified files to the staging area.

<file-path1> <file-path2> ... Add multiple files

-A Adds all changes, including modified, deleted, and new files, to the staging area.

git **commit** **-m** "first commit" 提交暂存区文件到 本地仓库内的当前分支目录，并添加备注（首次提交没有本地分支会自动创建 master 本地分支）【-am 无需 git add 直接提交】

git **push** **origin** dev 或 dev:fardev 上传本地分支 dev 合并到 远程分支 fardev（没有远程分支会自动新建，避免覆盖他人代码，先 pull 拉取远程仓库代码）【-u 追踪本地来源分支，一般用于首次提交】

git **push** **origin** :fardev 删除远程分支（默认分支无法删除，需要登录网页修改） 【--force 强制推送 --delete master 删除分支 】

git **push** **origin** dev:newdev 新建远程分支（上传本地分支） 【pull 中 --rebase 取消掉本地库中刚刚的 commit，并把他们接到更新后的版本库之中】

git pull origin <remote-branch>:<local-branch>

If there is not the default configured remote for your current branch, you must specify a branch on the command line.

git branch --set-upstream-to=<remote>/<remote-branch>

// git pull = git fetch + git merge FETCH_HEAD (FETCH_HEAD 指向拉取的远程分支最新的 commit，所以默认 pull 合并到当前分支)

--force 强行覆盖当前分支
--rebase 把 commit 取出来放到 stash，拉代码再 pop，等同 git fetch origin + git rebase develop

git pull <origin> dev 拉取 同名 追踪 远程分支 dev 合并到 本地分支 dev
git pull <origin> 拉取所有远程分支

git fetch <origin> 拉取 所有远程分支
git fetch <origin> fardev:temp 拉取 远程分支到本地（本地修改没有 stash 或者 commit 会导致 non-fast-forward）
git fetch <remote> <remote-branch>
This downloads the latest changes for the <remote-branch> branch from origin but doesn't affect your current branch.
The changes are stored in your local <remote>/<remote-branch> (the remote-tracking branch for dev).

git clone <repository-link> --recursive 递归的方式克隆整个项目
git submodule add <repository-link> <path> 添加子模块
// git submodule add git@github.com:saidake/simi-Initializer.git simi-plugin/simi-initializer
git submodule update 更新子模块
--remote Update all submodules from remote repository
git submodule init 初始化子模块
git submodule foreach git pull 拉取所有子模块
git rm <submodule-directory> Delete a submodule

git show Check git commit changes

git tag <tag_name> <commit_hash>
Tagging a Specific Commit
git tag -a <tag_name> -m "Tag message"
Annotated Tag
git tag -d <tag_name>
Delete a local tag
git push --delete origin <tag_name>
Delete a remote tag

git push origin <tag_name>
Pushing Tags to Remote Repository

git **cherry-pick** 0094ac868686 将其他分支上的 commit 修改内容合并到当前 branch【--abort 终止合并，并移除修改文件和新建 commit --continue 处理完 conflict 后，git add 再 continue 合并】
git **merge** origin FETCH_HEAD git **merge** origin develop 将其他分支合并到但概念分支（不会将新文件合并过来）
git **merge --no-ff --squash** new --no-ff main 跳到 new 后方创建新节点，main 收录 new 分支 --squash main 跳进到 new 后方创建新节点，但不和 new 连接
git reset HEAD~ 撤销当前分支 commit 提交

git update-index --assume-unchanged .classpath 忽略已跟踪文件的改动
git update-index --no-assume-unchanged /path/to/file 恢复跟踪
git revert HEAD 撤销前一次 commit
git revert HEAD^ 撤销前前一次 commit
git revert commit_id （比如:fa042ce57ebbe5bb9c8db709f719cec2c58ee7ff）

git rev-parse HEAD　获取 commitid
git checkout xxcommitid -b xxnewbranch
git **diff-tree** dev --stat　　比较两分支差异
git show --raw　查看最后一次提交记录的修改文件信息

修改文件　　文件被修改后，需要提交到某一分支，才能看到每个分支的内容
冲突处理　　两个分支都有修改--->冲突　默认修改的分支覆盖没有修改的分支
不同：（删除行内容也属于不同 mai --> mi）　　上方新增+不同　　下方新增+不同

ERROR　文件无法上传　　git 不支持空文件夹和 exe 文件的上传　　.gitignore 添加!*.exe

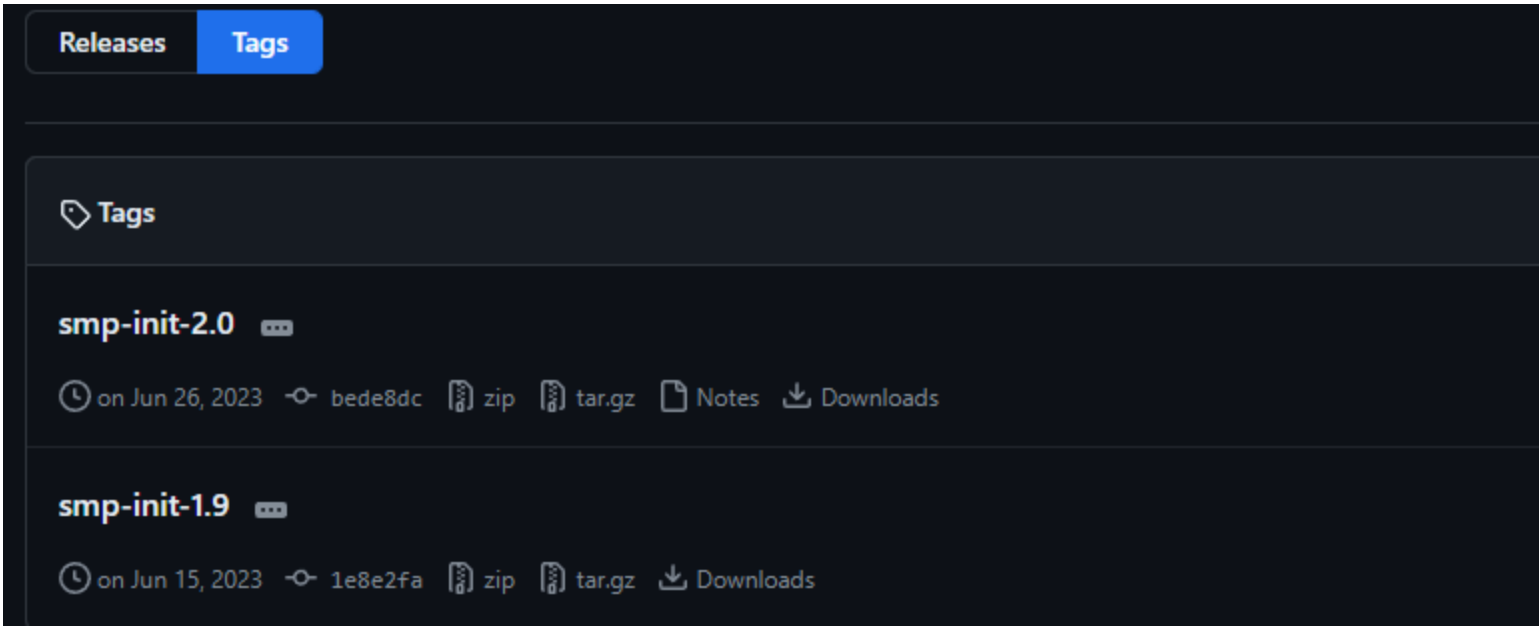
Git UI
Global

Default GitHub Commit View

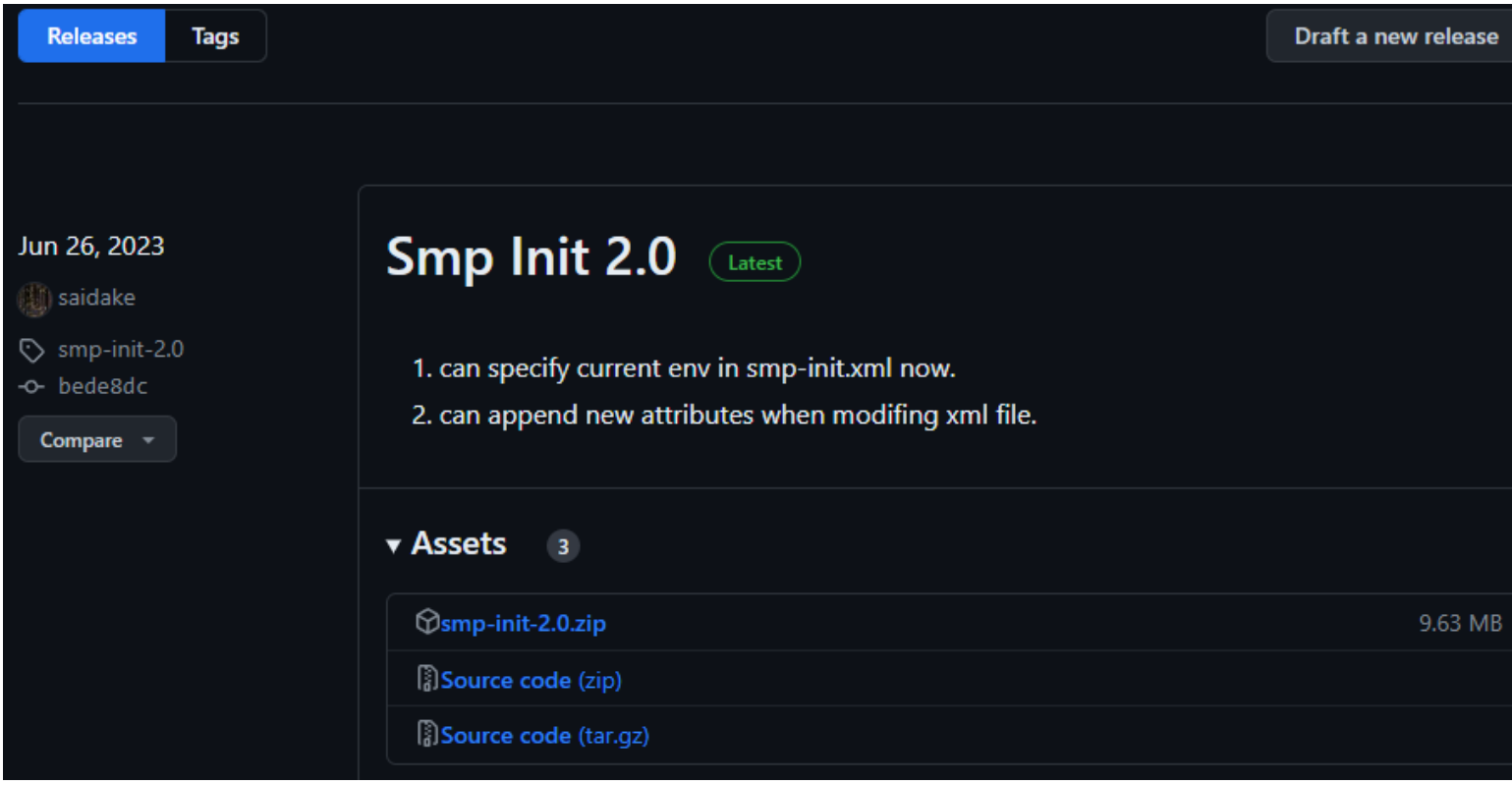
By default, GitHub's commit history view might hide merge commits to focus on individual commits and keep the history cleaner and more readable.

create release

Tags



Release title



gitLab

介绍

特征：私有 git 仓库，可以部署到自己的服务器上

开始：创建用户组，一个组里面可以有多个项目分支，可以将开发添加到组里面进行设置权限，访问项目

权限：Guest：可以创建 issue，发表评论，不能读写版本库

Reporter：可以克隆代码，不能提交，QA，PM 可以赋予这个权限

Developer：可以克隆代码，开发，提交，push，普通开发可以赋予这个权限

Maintainer：可以创建项目，添加 tag，保护分支，添加项目成员，编辑项目，核心开发可以赋予这个权限

Owner：可以设置项目访问权限，删除项目，迁移项目，管理组成员，开发组组长可以赋予这个权限。

安装

yum -y install policycoreutils openssh-server openssh-clients postfix

systemctl enable postfix && systemctl start postfix　　postfix 支持 gitlab 发信功能

firewall-cmd -add-service=ssh --permanent　　开放 ssh 以及 http 服务，然后重新加载防火墙列表

firewall-cmd -add-service=http --permanent

firewall-cmd --reload

apache-tomcat

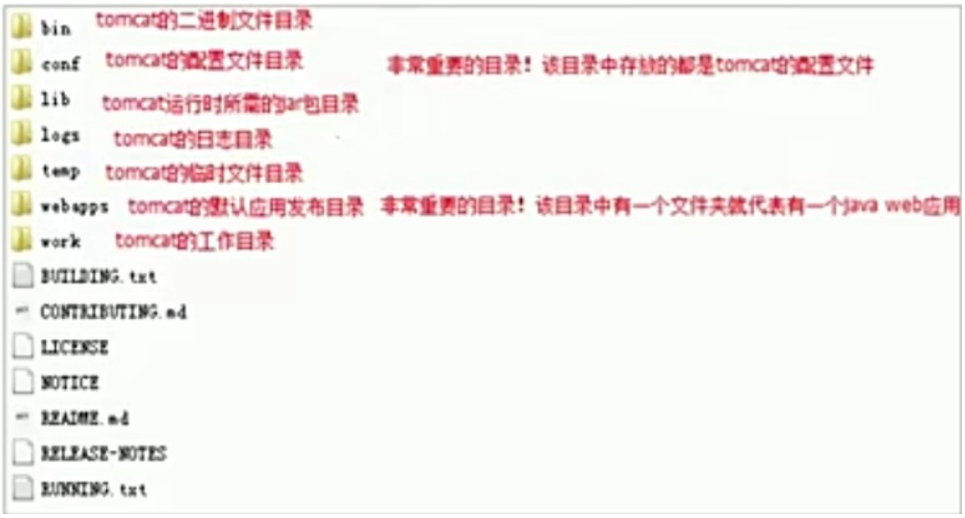
Core

介绍

win 安装　　bin > service.bat install 【位数必须与 java 一致，JAVA_HOME　JRE_HOME 必须有，http://localhost:8080/能够访问】

class 运行　　两个 tomcat 服务不能一起运行，只允许一个 class 文件夹配置项目运行（ 默认执行 java 工程路径：SMS\WebContent\WEB-INF\classes　）

linux 安装：　　startup.sh 启动



启动

startup.bat windows 下启动执行文件
startup.sh linux 下启动执行文件
shutdown.bat windows 下关闭执行文件
shutdown.sh linux 下关闭执行文件

配置

conf/server.xml >>

<Engine> 虚拟主机（可以指定访问路径名称）
<Host name="localhost" 访问域名（可添加多个 Host）
appBase="webapps" 项目存放路径
unpackWARs="true" 是否自动解压 war 包
autoDeploy="true"> 是否自动发布
<Context path="" 浏览器根目录前的访问目录
docBase="examsystem/WebContent" appBase 下的网站访问目录（默认加上 apBase 的路径，相对路径和绝对路径除外）
reloadable="true"/> 项目文件有改动时重新加载 （手动添加）
</Host>
<Connector port="**8080**" 访问端口
protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" **/>**
</Engine>

conf/web.xml >>

<servlet> 服务执行类
<init-param> 配置全局参数
<param-name>xxname<param- name >
<param-value>xxval<param-value>
</init-param> **</servlet>**
<welcom-file-list> 默认访问文件
<servlet> >>> <servlet-name> 类名 <servlet-class> 包名+类名
<servlet-mapping> 路由表 >>> <servlet-name> 类名 <url-pattern> URL 访问路由

Commands

httpd -v

httpd.conf

/opt/homebrew/etc/httpd/httpd.conf

python

python -m venv <path>
.\venv\Scripts\activate
deactivate
pip install -r <requirements-file>
pip install <python-module>

Linux 安装

1. 下载压缩包　　wget https://www.python.org/ftp/python/3.9.9/Python-3.9.9.tgz（检查新版本 https://www.python.org/downloads/source/）
2. 下载依赖包(必须)　　yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gcc make libffi-devel

wget　https://ftp.openbsd.org/pub/OpenBSD/LibreSSL/libressl-2.7.4.tar.gz　　libressl 依赖
./configure --prefix=usr/local　make　　make install　　安装
vi /etc/ld.so.conf.d/local.conf　　修改或新建库配置文件
ldconfig -v　　重新加载共享模块

创建安装目录　　mkdir -p /usr/local/python3
解压　　tar -zxvf Python-3.7.1.tgz
进入解压文件夹配置安装目录　　./configure --prefix=/usr/local/python3　　生成 makefile
编译　　make
openssl 安装

wget https://www.openssl.org/source/openssl-1.1.1a.tar.gz
tar -zxvf openssl-1.1.1a.tar.gz
cd openssl-1.1.1a　编译安装
./config --prefix=/usr/local/openssl no-zlib #不需要 zlib
make && make install
sudo mv /usr/bin/openssl /usr/bin/openssl.bak
sudo ln -sf /usr/local/openssl/bin/openssl /usr/bin/openssl
sudo vim /etc/ld.so.conf 打开文件，把/usr/local/openssl/lib 加的文件最后：
ln -s /usr/local/lib/libssl.so.1.1 /usr/lib/libssl.so.1.1
ln -s /usr/local/lib/libcrypto.so.1.1 /usr/lib/libcrypto.so.1.1

编辑　Modules/Setup　解除 ssl 注释
SSL=/usr/local
_ssl _ssl.c \
-DUSE_SSL -I$(SSL)/include -I$(SSL)/include/openssl \
-L$(SSL)/lib -lssl -lcrypto

清空编译文件 make clean && make distclean
安装　　make && make install
添加环境变量　　/etc/profile　　export PATH=$PATH:/home/saidake/userprogram/python3/bin

pip 镜像：pip config set global.index-url https://pypi.douban.com/simple/

chrome 安装：　　yum install https://dl.google.com/linux/direct/google-chrome-stable_current_x86_64.rpm

yum install mesa-libOSMesa-devel gnu-free-sans-fonts wqy-zenhei-fonts
wget http://npm.taobao.org/mirrors/chromedriver/2.41/chromedriver_linux64.zip
unzip chromedriver_linux64.zi

pip 安装

set PYTHONIOENCODING=utf8　设置 python 字符串编码
pip freeze　查看包版本
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple/
pip list　查看安装的包 【--outdated　列出可以更新的包】
pip uninstall xxx　卸载包
pip install --upgrade pip　更新包　// EX：pip install redis==2.10.6　指定版本
pip install pipreqs　依赖管理
pipreqs ./ --encoding utf-8 --force　自动生成 requirements.txt 【--encoding utf-8　设置编码　--force 覆盖】
pip install -r requirements.txt　安装依赖
python -m ensurepip　确保 pip 已经下载安装 【-u 参数后会强制其标准输出也同标准错误一样不通过缓存直接打印到屏幕】

anacoda

1. 官网安装
2. conda --version　查看版本
3. pip install tensorflow　matplotlib

venv 虚拟环境

python -m venv xxname　创建虚拟环境　// python -m venv venv
./activate.bat　进入虚拟环境
./ deactivate.bat　推出虚拟环境

linux venv 虚拟环境

pip3 install virtualenv
virtualenv xxname　【--no-site-packages　创建一个干净隔离的 python 虚拟环境　--python=python 】
source /xxx/virtualname/bin/active　进入虚拟环境，路径为创建虚拟环境时的路径
deactivate　直接执行命令。退出当前虚拟环境

pm2 (Node Process Management)

安装

npm install -g pm2　确保有 node.js 服务

命令

pm2 start app.js --name xxx　启动并命名进程
pm2 start app.js -i 4　后台运行 pm2，启动 4 个 app.js
也可以把'max' 参数传递给 start
正确的进程数目依赖于 Cpu 的核心数目

pm2 list　显示所有进程状态
pm2 monit　监视所有进程
pm2 logs　显示所有进程日志
pm2 stop all　停止所有进程
pm2 restart all　重启所有进程
pm2 reload all　0 秒停机重载进程 (用于 NETWORKED 进程)
pm2 stop 0　停止指定的进程
pm2 restart 0　重启指定的进程
pm2 startup　产生 init 脚本 保持进程活着

pm2 web　　运行健壮的 computer API endpoint (http://localhost:9615)
pm2 delete 0　　杀死指定的进程
pm2 delete all　　杀死全部进程

配置项

pm2 deploy ecosystem.json setup　&&pm2 deploy ecosystem.json　　自动部署
(本地和服务器上同时安装好 pm2 、git， 本地 pm2 向 git 提交代码， 同时服务器的 PM2 拉取最新的代码，并在拉取成功后运行代码)
保证服务器和本地的 git 都能正常 clone，并且两个 rsa 都添加到 git 上去了，权限为 700

ecosystem.json >

```
{
    "apps": [{
        "name": "ice",                  //对应 Nginx 上的配置
        "script": "server.js",          //用来启动的脚本
        "watch": true,
        "instances": max,
        "exec_mode": "cluster",
        "max_memory_restart": ""
        "env": {                                //启动传入项目的环境变量
            "COMON_VARIABLE": "true"
        },
        "env_production": {             //部署环境
            "NODE_ENV": "production"
        }
     }],
    "deploy": {
        "production": {
            "user": "root",
            "host": ["101.132.109.40"],                                 //服务器 ip 地址
            "port": "22",                                                   //登录端口
            "ref": "origin/master",                                         //拉取对应分支的代码
            "repo": "git@gitee.com:qjnugede/ice.git",           //仓库地址
            "path": "/www/ice/production",                              //要部署到服务器哪个位置，如果没有会自动创建
            "ssh_options": "StrictHostKeyChecking=no",          //避免 key 验证导致代码更新到远程仓库失败
            "post-deploy": "npm install --registry=https://registry.npm.taobao.org && grunt build && pm2 startOrRestart ecosystem.json --env production",
                                                                                                    //发布之后执行的动作 执行开启或更新 pm2 运行的服务
            "pre-deploy-local":"echo 'Deploy Done!'",           //本地发布之前的动作
            "env": {
                "NODE_ENV": "production"                                //指定部署到远程的仓库的环境 是 production 生产环境
            }
     }}
}
```

server.js >

```
const http = require('http');
```

```
const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello finaish\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

pm2 deploy ecosystem.json production

Mermaid

UML Class Diagram

A **class diagram** represents the structure of a system by showing classes, their attributes, methods, and relationships. The key elements include:

Classes

```
classDiagram
class BankAccount{
      +String owner
      +BigDecimal balance
      +deposit(amount: String) bool
      +withdrawal(amount: String) int
}
```

Generic Types

```
classDiagram
class Square~Shape~{
      int id
      List~int~ position
      setPoints(List~int~ points)
      getPoints() List~int~
}

Square : -List~string~ messages
Square : +setMessages(List~string~ messages)
Square : +getMessages() List~string~
Square : +getDistanceMatrix() List~List~int~~

* Abstract e.g.: someAbstractMethod()* or someAbstractMethod() int*
$ Static e.g.: someStaticMethod()$ or someStaticMethod() String$
```

Relationships

```
Association ( --> )
      Shows a connection between classes.
Dependency ( ..> )
      Indicates one class depends on another.
Aggregation ( o-- )
```

A "whole-part" relationship where parts can exist independently.

Composition (*--)

A "whole-part" relationship where parts cannot exist independently.

Inheritance (Generalization) (<|--)

Represents an "is-a" relationship.

Realization (Interface Implementation) (..|>)

Indicates a class implements an interface.

Link (Dashed) (..)

Link (Solid) (--)

Two-way relations

[Relation Type][Link][Relation Type]

```
classDiagram
    Animal <|--|> Zebra
```

Visibility Modifiers

+ (Public)

Accessible from anywhere.

(Protected)

Accessible within the class and its subclasses.

- (Private)

Accessible only within the class.

~ (Package)

Accessible within the same package.

Flow Diagram

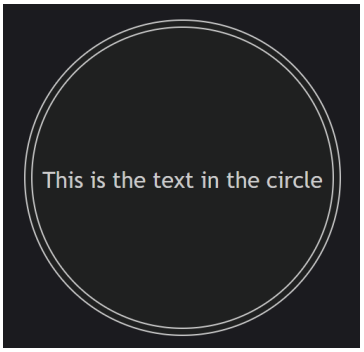
Rectangle

```
flowchart LR
    id1[[This is the text in the box]]
```

Circle

```
flowchart LR
    id1((This is the text in the circle))
```

```
flowchart TD
    id1(((This is the text in the circle)))
```



Event

```
flowchart TD
    A@{ shape: rounded, label: "This is an event" }
```

Terminal Point (Stadium)

```
flowchart TD
    A@{ shape: stadium, label: "Terminal point" }
```

A stadium-shaped node

```
flowchart LR
    id1([This is the text in the box])
```

Subgraph

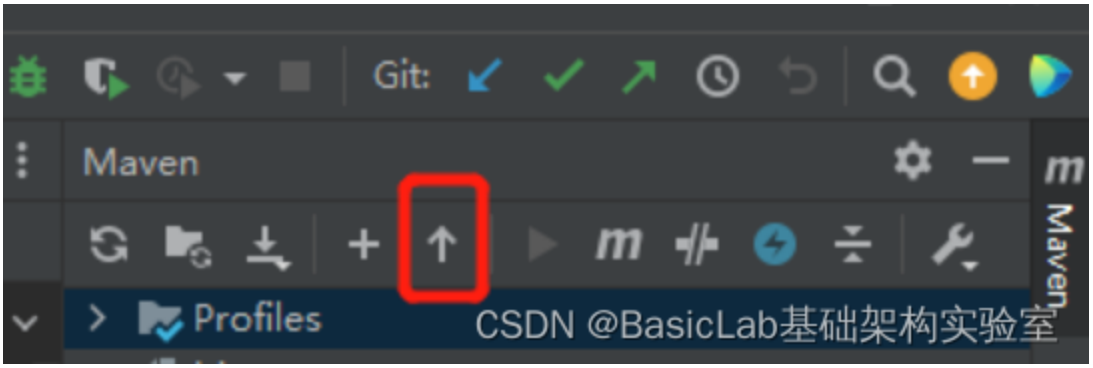
```
flowchart TB
    c1-->a2
    subgraph one
    a1-->a2
    end
    subgraph two
    b1-->b2
    end
    subgraph three
    c1-->c2
    end

flowchart LR
  subgraph TOP
    direction TB
    subgraph B1
        direction RL
        i1 -->f1
    end
    subgraph B2
        direction BT
        i2 -->f2
    end
  end
  A --> TOP --> B
  B1 --> B2
```

Config / Development

intellij IDEA

Plugin

Maven Project Version　　多级修改 maven 项目版本



python community edition（添加 python）　Free Mybatis plugin　maven-helper（检查包冲突）
google-java-format
Material Theme UI　界面主题　Atom Meterial Icon　图标主题（设置搜索 closure 关闭折叠按钮）
Restful Fast Request　类似 postman
PlantUML　绘制思维导图
GsonFormatPlus　根据 json 生成实体类
Maven Helper　解决依赖冲突
MarkDown　md 文件预览　Preferences --> Editor --> File Types --> Markdown
arthas idea　基于 IDEA 开发的 Arthas 命令生成插件，支持 Arthas 官方常用的命令，比如 watch、trace、ognl static、ognl bean method、field、monitor、stack 、tt 等命令。
JMH Java Microbenchmark Harness　Harness test

Interface

禁用 ideaVIM
快捷键
inspection　spelling　typo　关闭拼写检查
File　Invalideate Caches / Restart　清除缓存并重启（解决类变红问题）
Settings -> Editor -> Live Templates　定义代码块（1　变量）

Shortcut

Alt + 2 or Cmd + 2 (on macOS)　Open the Bookmarks tool window.
Ctrl N or Cmd O (on macOS)　Find a class

ctrl LeftMouse　查看源码
ctrl alt 左键　查看实现类
ctrl E　编译（手动）

ctrl shift R　全局搜索字符串
ctrl L　搜索类

alt 7　显示类结构
alt F7　显示调用路线
ctrl H　查询父子结构
ctrl shift alt U　显示依赖结构
alt insert　生成 getter 等方法
ctrl alt M　抽取方法

alt Enter　自动导入

alt J 选中下一个相同的

ctrl alt t 快速生成代码
ctrl alt l 格式化
ctrl G 跳转到行数

ctrl alt b 显示源码
ctrl O 显示实现方法
ctrl insert 显示实现方法
ctrl alt + 展开所有方法
ctrl shift + 展开所有
ctrl shift - 收缩所有

alt F1 打开文件所在位置

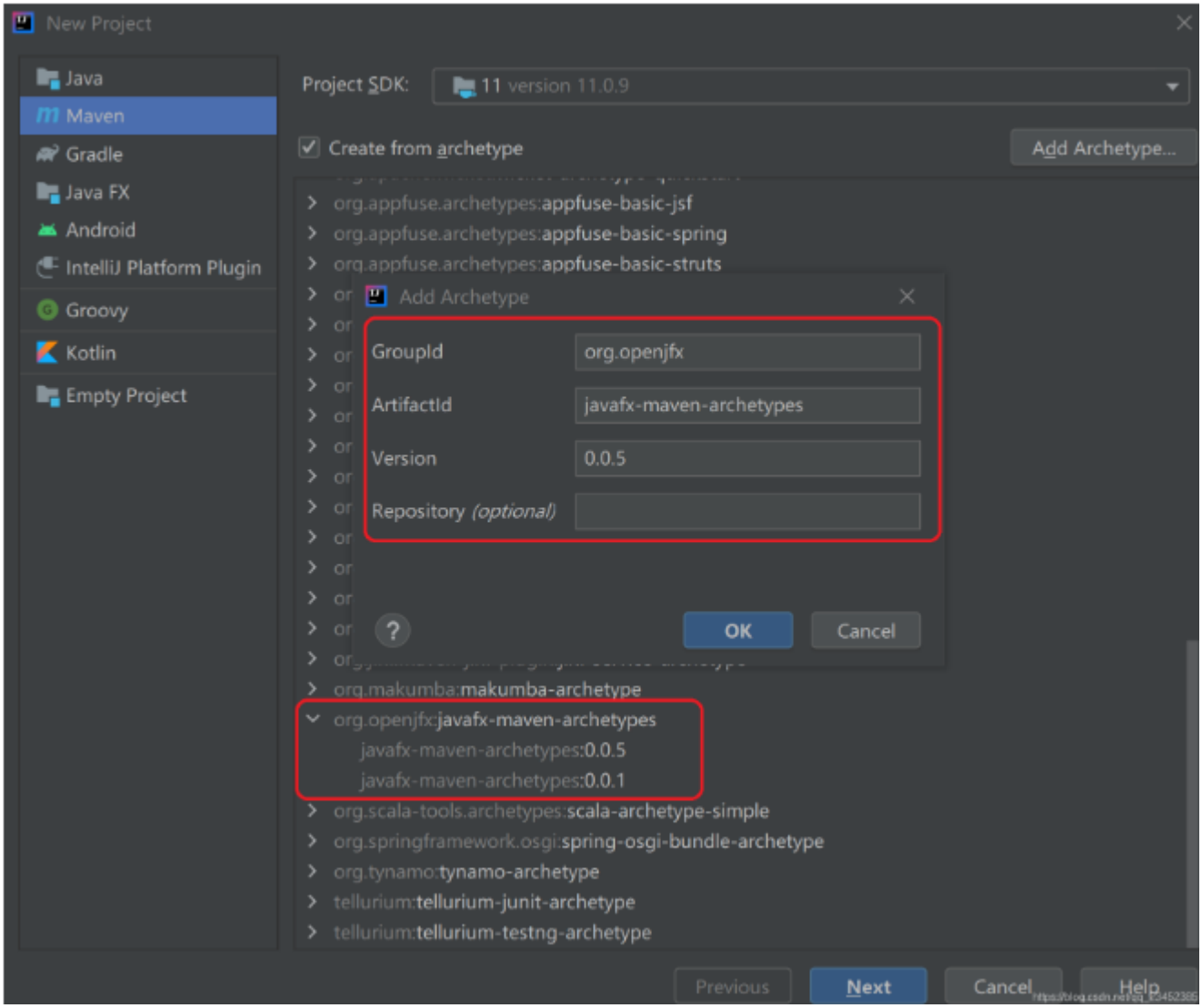
ERROR

Can't load all modules
    Import the parent module
Build success but can't find package in a specific module
    Jump to the module in terminal, rebuild this module and reload this module0
JDK isn't specified for module "xxx"
    re-Import the parent module

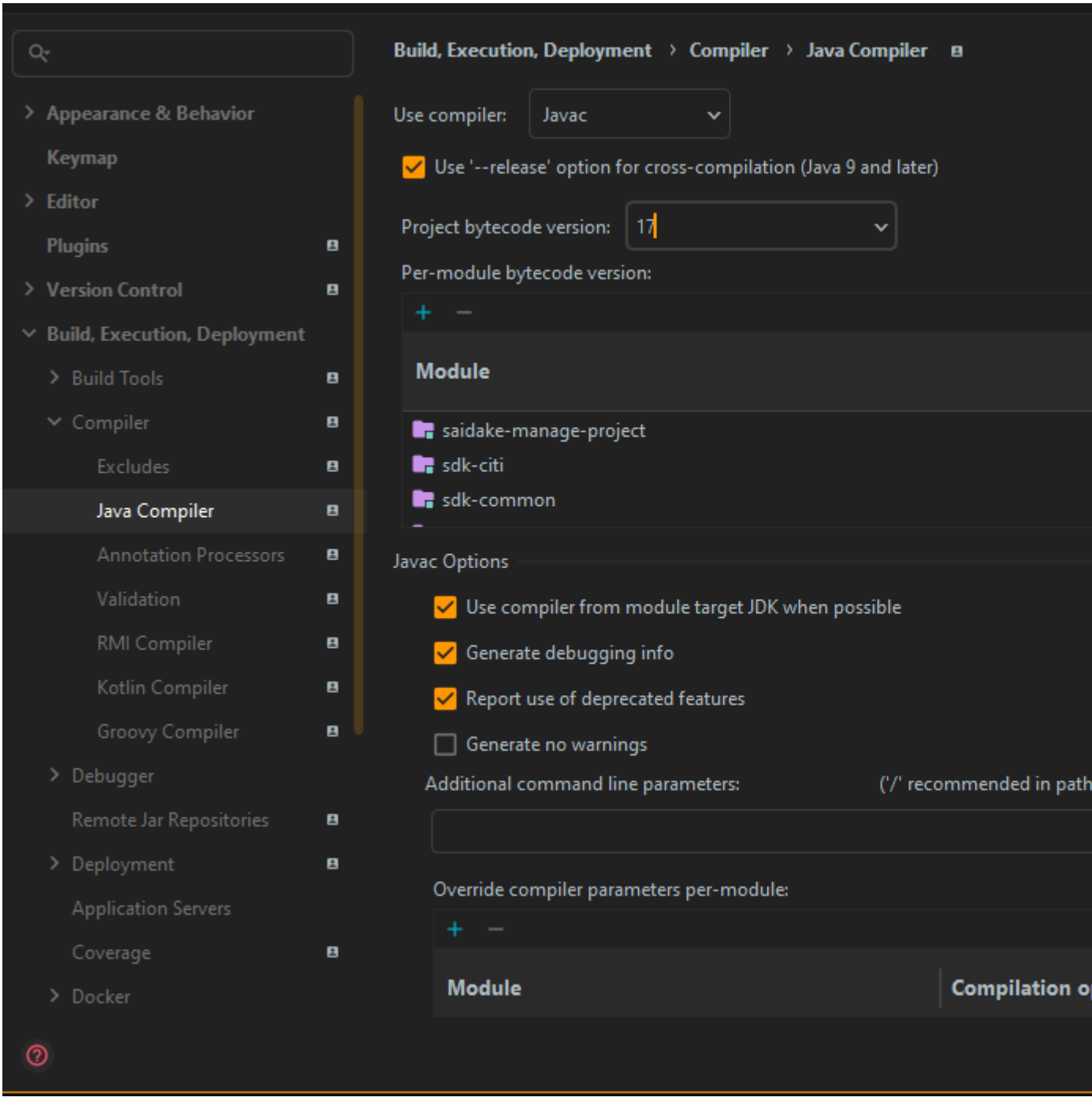
创建项目

archetype 模板

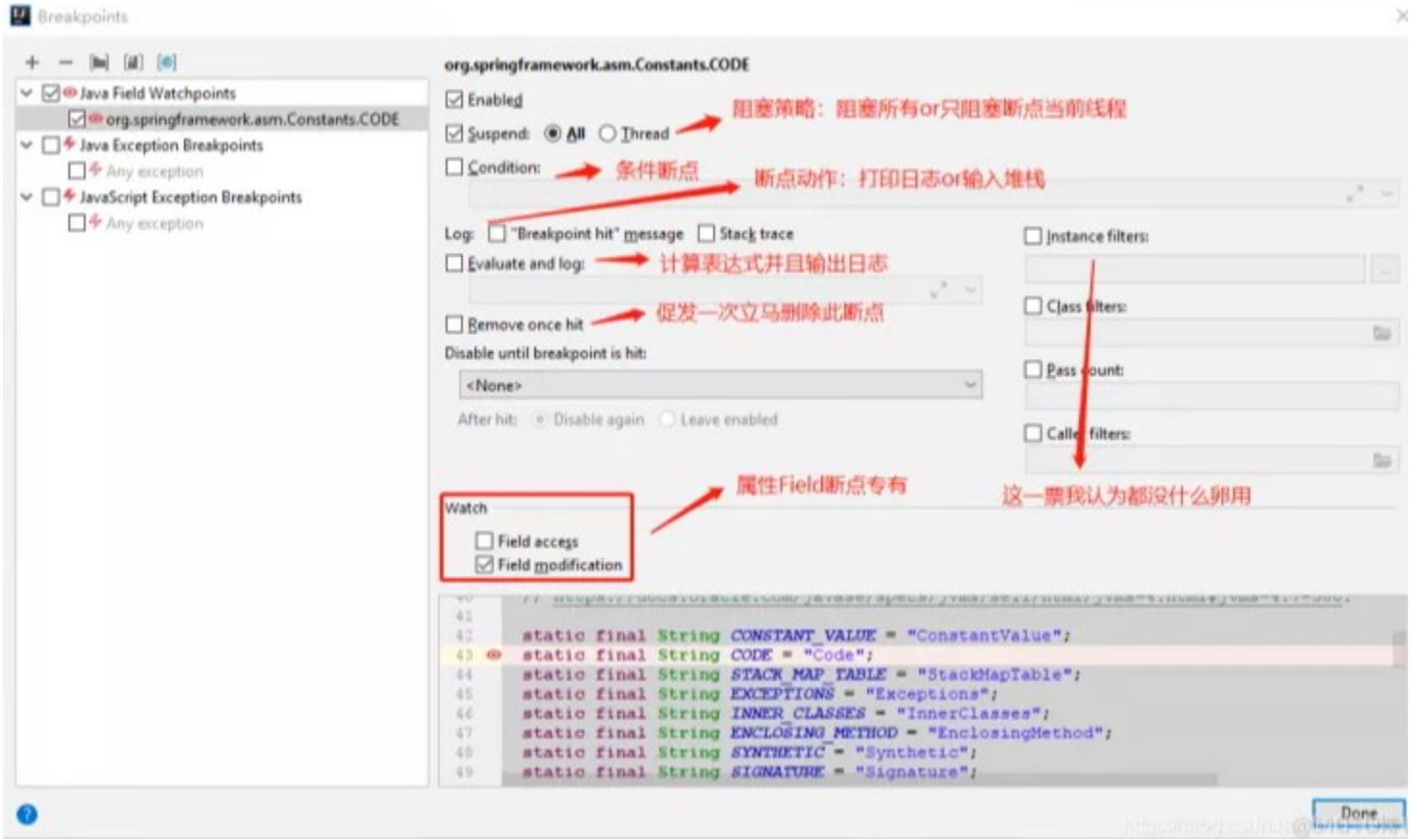
org.apache.cocoon:cocoon-22-archetype-webapp 创建 webapp 模块（带有 WEB-INF 目录）


New Project
Java
Maven
Gradle
Java FX
Android
IntelliJ Platform Plugin
Groovy
Kotlin
Empty Project
Project SDK:
11 version 11.0.9
Create from archetype
Add Archetype...
org.appfuse.archetypes:appfuse-basic-jsf
org.appfuse.archetypes:appfuse-basic-spring
org.appfuse.archetypes:appfuse-basic-struts
Add Archetype
GroupId
org.openjfx
ArtifactId
javafx-maven-archetypes
Version
0.0.5
Repository (optional)
OK
Cancel
org.makumba:makumba-archetype
org.openjfx:javafx-maven-archetypes
javafx-maven-archetypes:0.0.5
javafx-maven-archetypes:0.0.1
org.scala-tools.archetypes:scala-archetype-simple
org.springframework.osgi:spring-osgi-bundle-archetype
org.tynamo:tynamo-archetype
tellurium:tellurium-junit-archetype
tellurium:tellurium-testng-archetype
Previous
Next
Cancel
Help

项目字节码版本


Build, Execution, Deployment › Compiler › Java Compiler
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Build Tools
Compiler
Excludes
Java Compiler
Annotation Processors
Validation
RMI Compiler
Kotlin Compiler
Groovy Compiler
Debugger
Remote Jar Repositories
Deployment
Application Servers
Coverage
Docker
Use compiler: Javac
Use '--release' option for cross-compilation (Java 9 and later)
Project bytecode version: 17
Per-module bytecode version:
Module
saidake-manage-project
sdk-citi
sdk-common
Javac Options
Use compiler from module target JDK when possible
Generate debugging info
Report use of deprecated features
Generate no warnings
Additional command line parameters:
('/' recommended in path
Override compiler parameters per-module:
Module
Compilation o

调试



Class Filter： com.saidake.citi.* java.util.* 只拦截 com.saidake.citi 和 java.util 包下 所有类的异常

*s2 拦截 s2 结尾的类

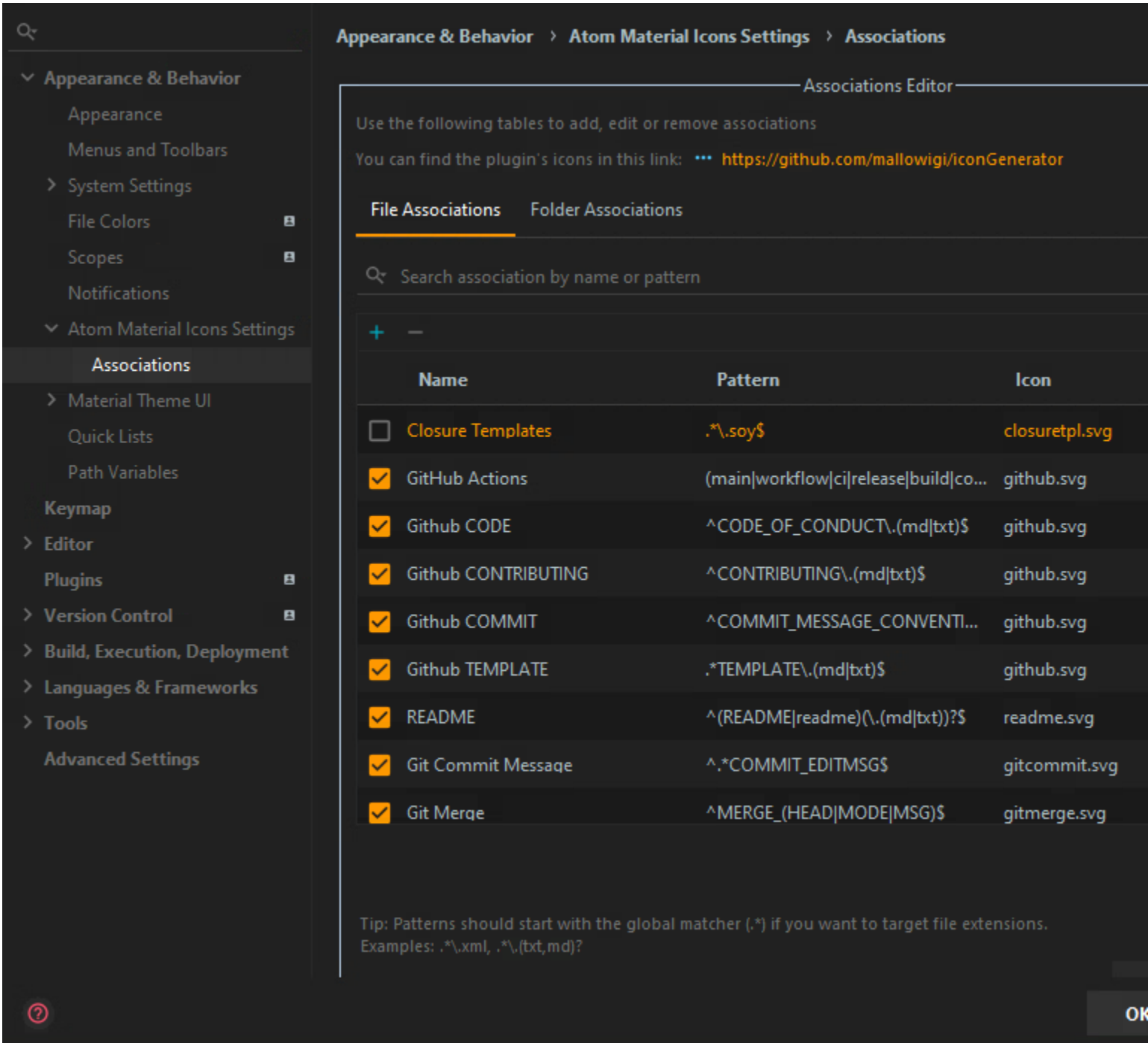
-java.* -sun.* 排除 java 包下的所有类

-package1.Class1 排除 Class1 类

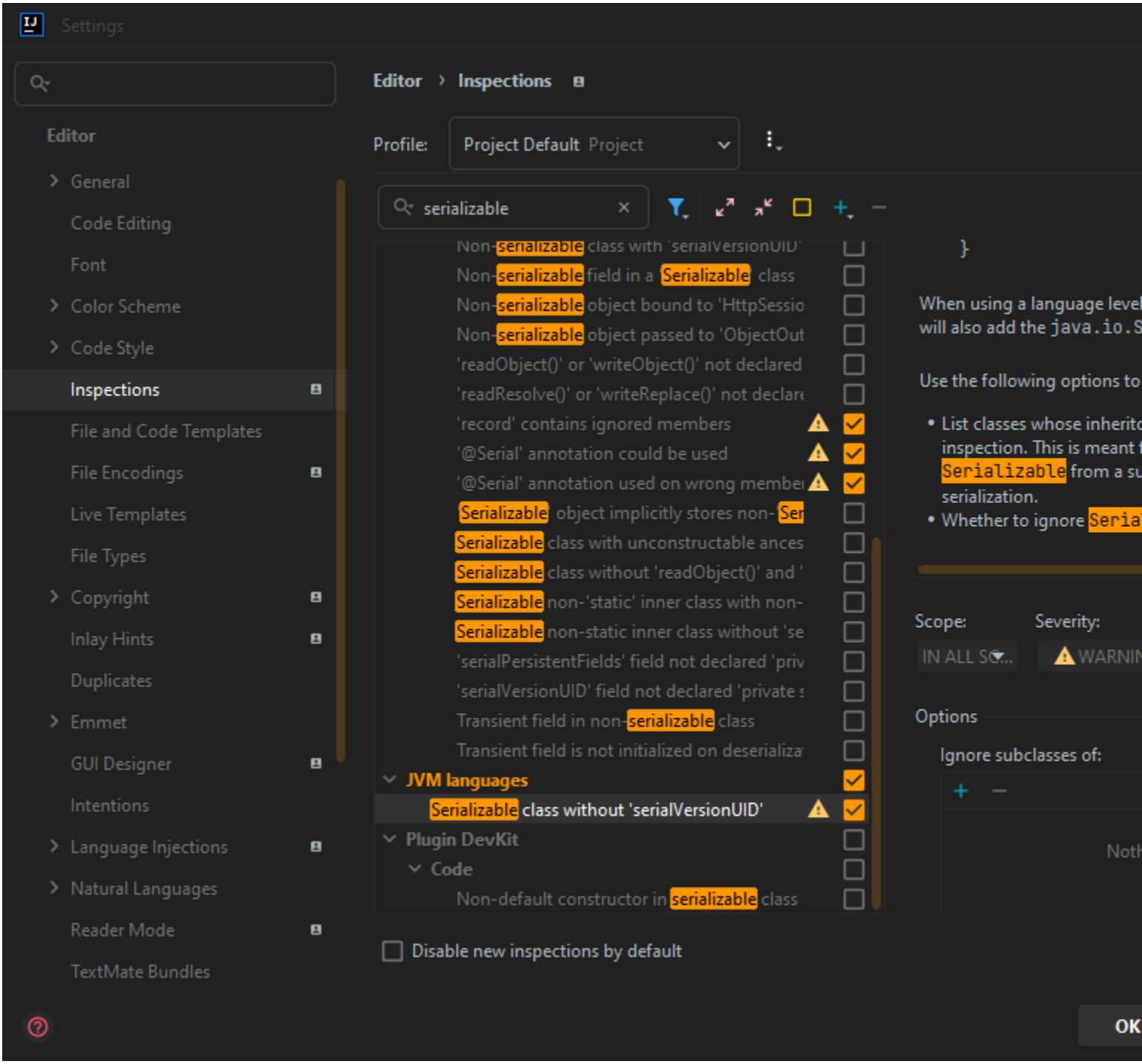
Caller Filter： -package1.Class1.method1([Ljava/lang/String;)Ljava/util/List; 排除方法 List method1 (String[] input)

初始功能配置

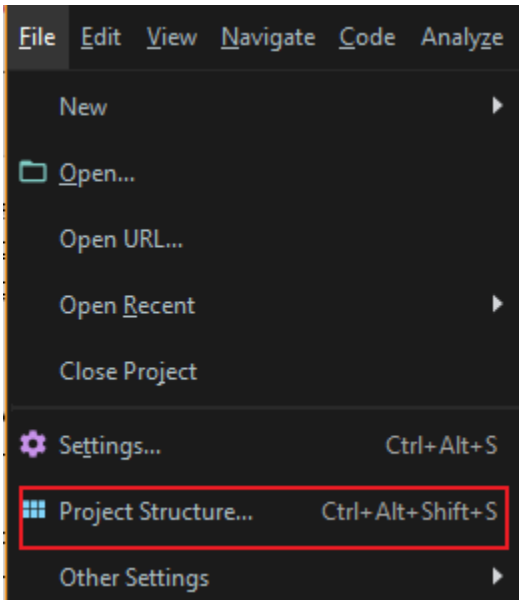
去掉文件夹下箭头 ui


Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Atom Material Icons Settings
Associations
Material Theme UI
Quick Lists
Path Variables
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
Appearance & Behavior › Atom Material Icons Settings › Associations
Associations Editor
Use the following tables to add, edit or remove associations
You can find the plugin's icons in this link: https://github.com/mallowigi/iconGenerator
File Associations
Folder Associations
Search association by name or pattern
Name
Pattern
Icon
Closure Templates .*\.soy$ closuretpl.svg
GitHub Actions (main|workflow|ci|release|build|co... github.svg
Github CODE ^CODE_OF_CONDUCT\.(md|txt)$ github.svg
Github CONTRIBUTING ^CONTRIBUTING\.(md|txt)$ github.svg
Github COMMIT ^COMMIT_MESSAGE_CONVENTI... github.svg
Github TEMPLATE .*TEMPLATE\.(md|txt)$ github.svg
README ^(README|readme)(\.(md|txt))?$ readme.svg
Git Commit Message ^.*COMMIT_EDITMSG$ gitcommit.svg
Git Merge ^MERGE_(HEAD|MODE|MSG)$ gitmerge.svg
Tip: Patterns should start with the global matcher (.*) if you want to target file extensions.
Examples: .*\.xml, .*\.(txt,md)?
OK

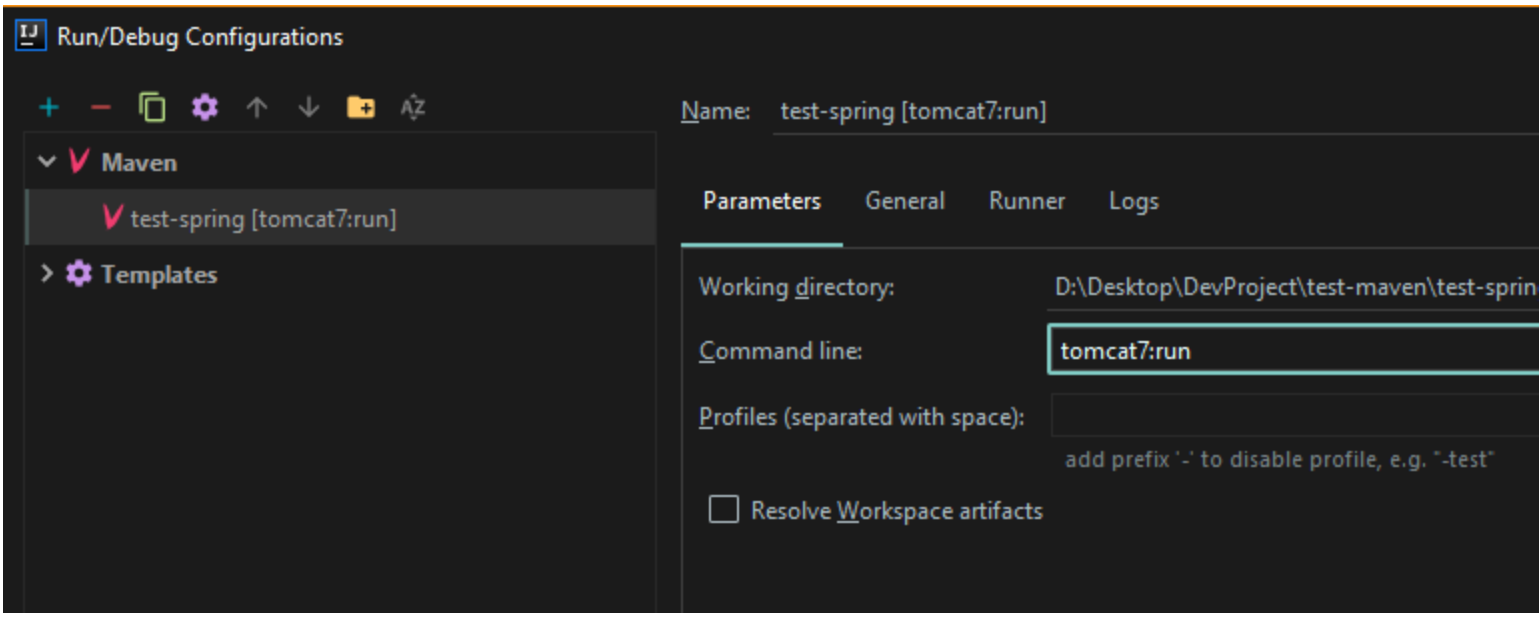
自动生成 uuid


Settings
Editor
General
Code Editing
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Copyright
Inlay Hints
Duplicates
Emmet
GUI Designer
Intentions
Language Injections
Natural Languages
Reader Mode
TextMate Bundles
Editor › Inspections
Profile: Project Default Project
serializable
Non-serializable class with 'serialVersionUID'
Non-serializable field in a Serializable class
Non-serializable object bound to 'HttpSessio
Non-serializable object passed to 'ObjectOut
'readObject()' or 'writeObject()' not declared
'readResolve()' or 'writeReplace()' not declare
'record' contains ignored members
'@Serial' annotation could be used
'@Serial' annotation used on wrong member
Serializable object implicitly stores non-Ser
Serializable class with unconstructable ances
Serializable class without 'readObject()' and '
Serializable non-'static' inner class with non-
Serializable non-static inner class without 'se
'serialPersistentFields' field not declared 'priv
'serialVersionUID' field not declared 'private s
Transient field in non-serializable class
Transient field is not initialized on deserializa
JVM languages
Serializable class without 'serialVersionUID'
Plugin DevKit
Code
Non-default constructor in serializable class
Disable new inspections by default
}
When using a language level
will also add the java.io.S
Use the following options to
List classes whose inherito
inspection. This is meant f
Serializable from a su
serialization.
Whether to ignore Seria
Scope:
Severity:
IN ALL SC...
WARNIN
Options
Ignore subclasses of:
Noth
OK

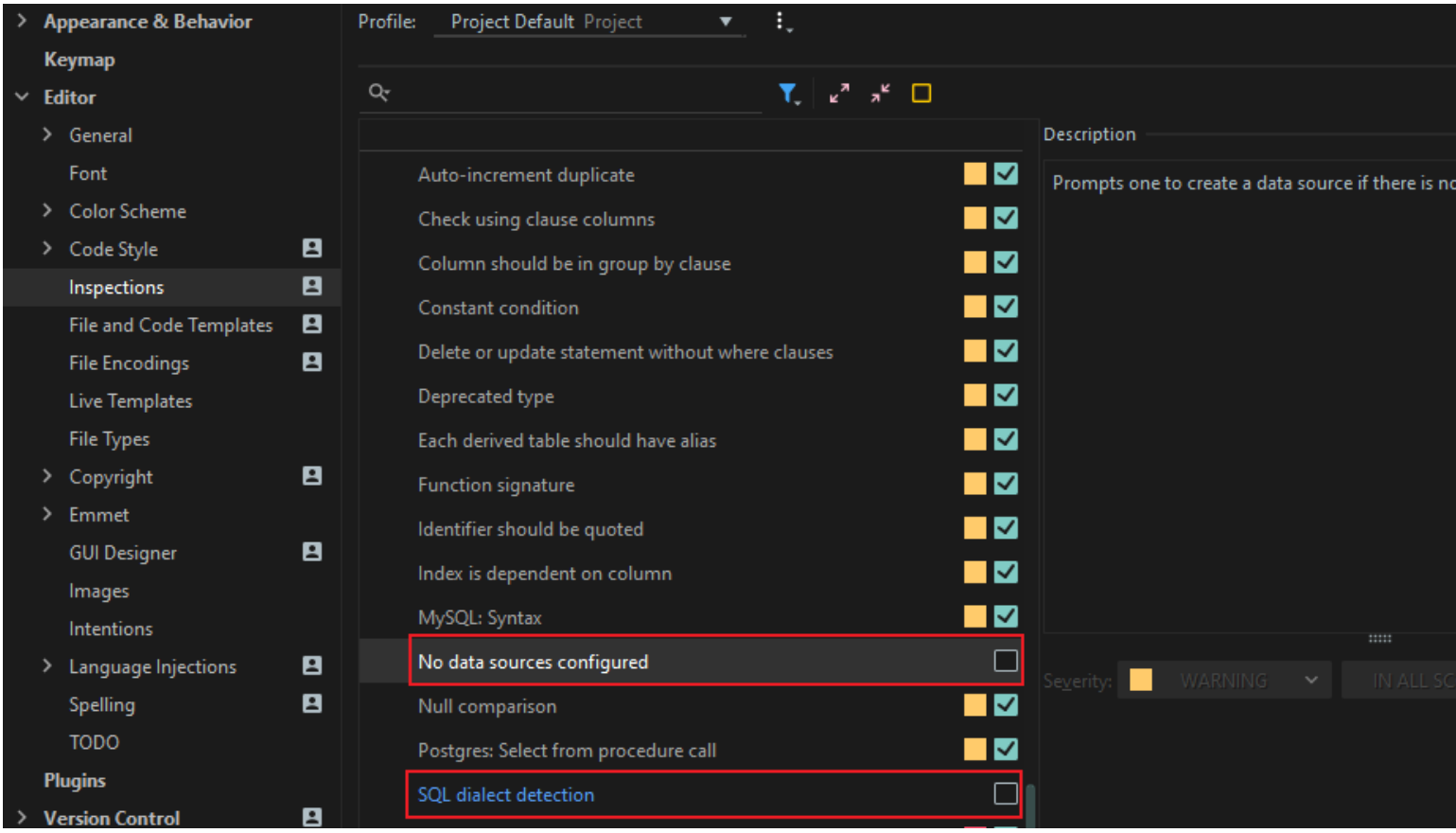
配置项目 sdk



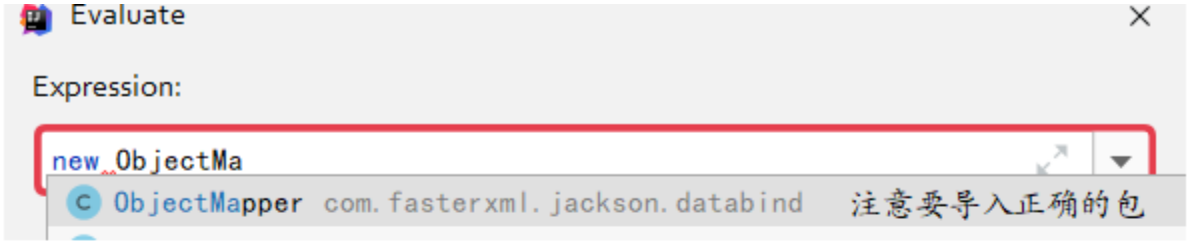
创建快捷启动

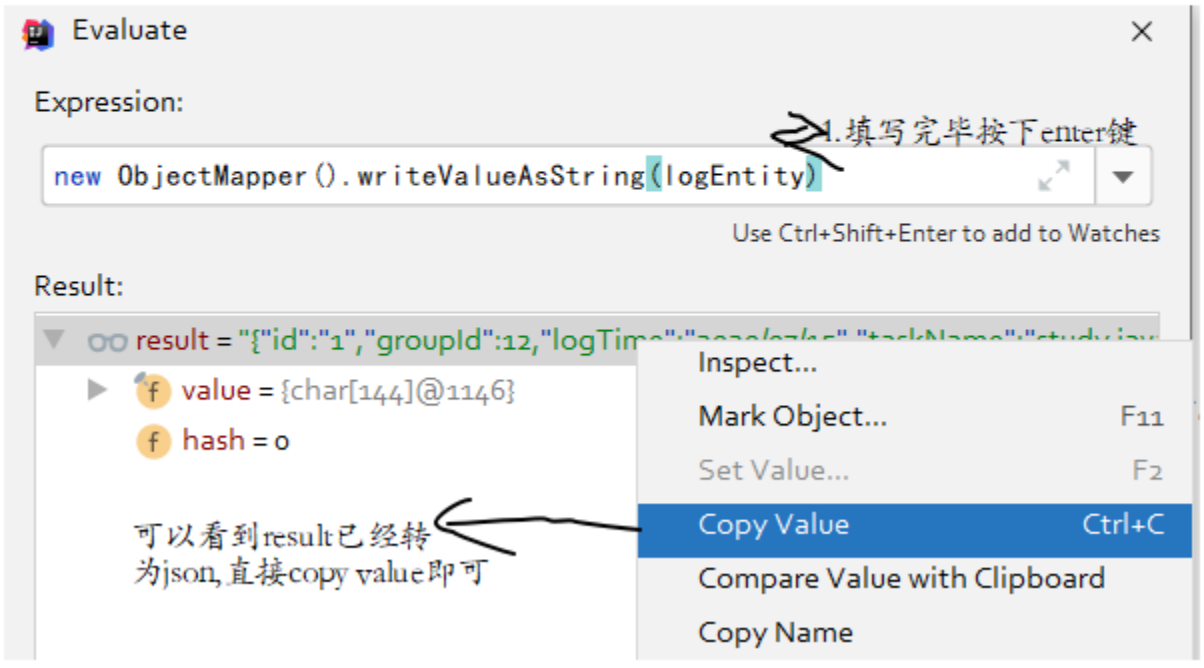


mybatis 去除下划线，SQL 检测配置



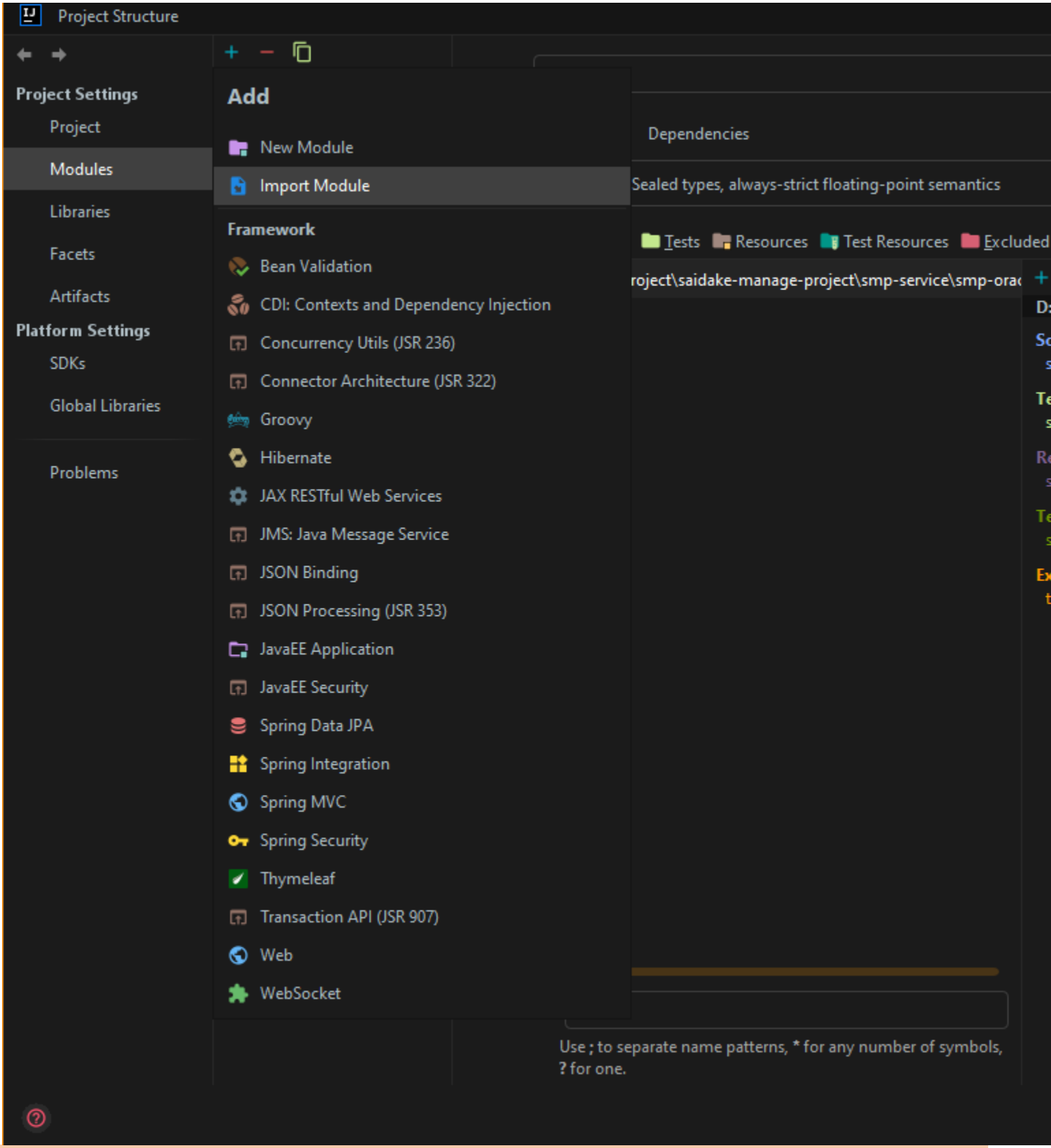
断点数据转 json





module

重命名后重新导入 module



其他

ERROR 大量红色类: .idea 删除 vcs File invalidate cache

无法指定主类: file–project structure–Modules 右键将 src 文件夹设置为 Sources

could not find or load main class 重新 compile

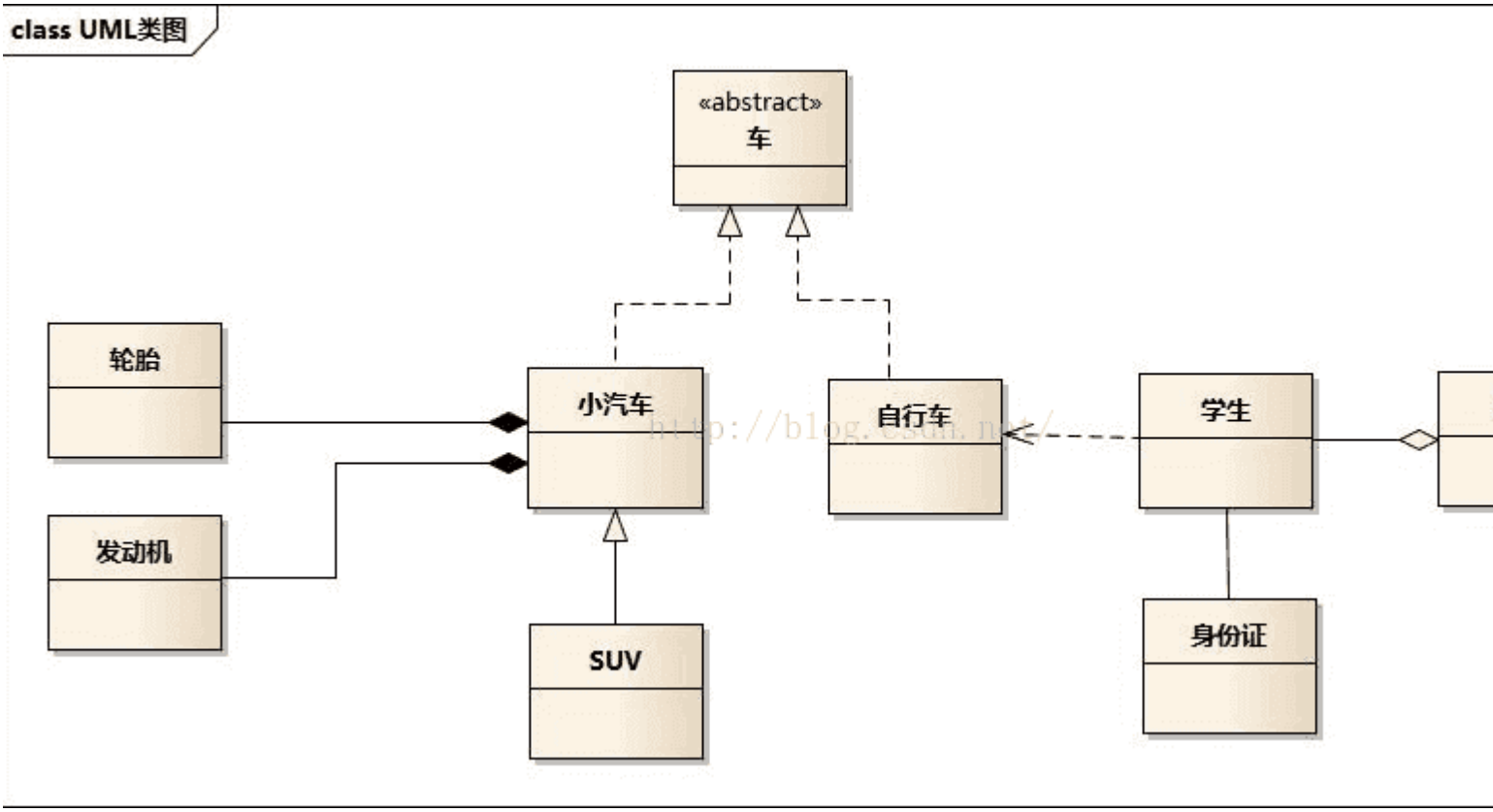
lombok 报错：　　下载 lombok idea 插件

依赖红色波浪线：　　删除.m2 目录下的错误 jar 包，刷新重新导入

项目 Structure 修改：

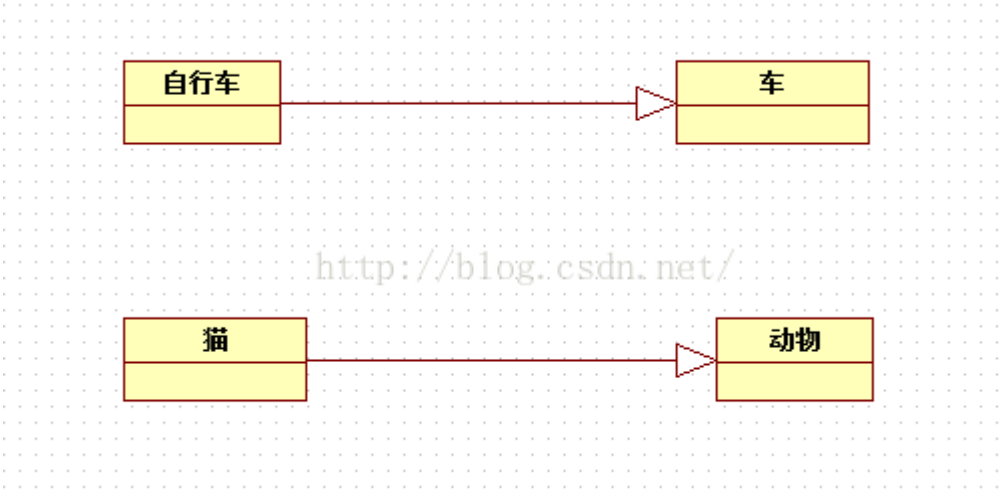
SourceFolders	src\main\java
TestSourceFolders	src\test\java
Resource Folders	src\main\resources
Excluded Folders	target

StartUML



Generalization：

泛化关系：类的继承 is-a 关系，两个对象之间如果可以用 is-a 来表示，就是继承关系。泛化关系用一条带空心箭头的直接表示。



Realization：

实现关系：用一条带空心箭头的虚线表示。

Dependency：

依赖关系：用一套带有箭头的虚线表示，带箭头的一端表示被依赖，他描述一个对象在运行期间会用到另一个对象。

与关联关系不同的是，它是一种临时性的关系，通常在运行期间产生，并且随着运行时的变化，依赖关系也可能发生变化。

Assocaition 和 DirectedAssocaition:

关联关系：关联关系是用一条直线表示的，它描述不同类的对象之间的结构关系，它是一种静态关系，通常与运行状态无关，一般由常识等因素决定的。

它是一种“强联”的关系.关联关系默认不强调方向，表示对象间相互知道，如果特别强调方向，则使用箭头。

Aggregation:

聚合关系：聚合关系是用一条带空心菱形的直线表示，用于表示实体对象之间的关系，表示整体由部分构成。

Composition:

组合关系：用一条带实心菱形箭头直线表示。菱形所在的一端代表主体，表示前者组成了主体，或者主体由其他的组成。

Sublime3176

安装

127.0.0.1 www.sublimetext.com
127.0.0.1 license.sublimehq.com

1. 添加 Package Control.sublime-package 到 Browse Packages 选项 Installed Packages 目录
2. package Control: install Package

3. 添加目录打开：添加目录 HKEY_CLASSES_ROOT\Directory\shell\1-sublime\command
   添加主题 1-sublime default Sublime Open Folder
   icon D:\UserProgram\Sublime Text 3\sublime_text.exe
   添加执行值 command default "D:\UserProgram\Sublime Text 3\sublime_text.exe" "%1"
4. 添加文件打开：添加目录 HKEY_CLASSES_ROOT*\shell\1-sublime\command
   添加主题 1-sublime default Sublime Open File
   icon D:\UserProgram\Sublime Text 3\sublime_text.exe
   添加执行值 command default "D:\UserProgram\Sublime Text 3\sublime_text.exe" "%1"
5. 绑定文件后缀： 修改 txt 打开方式 HKEY_CLASSES_ROOT\txtfile\shell\open\command "D:\UserProgram\Sublime Text 3\sublime_text.exe" "%1"

插件

Manzhou 主界面
Enki Alt 文件界面
A File Icon 图标

Emmet html 补全
> 下一个子标签
* 乘多少个
$ 标签序号
{} 标签内容 ul>li{我是第$个}*10
input:button 表单类型
div.test.aa.bb 添加多个类
CodeFormatter PHP, js, json, html, css, python 美化源码【修改 Packages\ CodeFormatter [{ "keys": ["ctrl+r"], "command": "color_formatter" }] 】
SublimeAStyleFormatter 格式化工具
SublimeJavaImports java 自动导入包

DocBlockr 注释插件 （在方法前输入/** 回车）

ColorPicker 颜色拾取器 【修改 Packages\Terminal [{ "keys": ["ctrl+e], "command": "color_pick" }] 】
Color Highlighter 颜色高亮
AutoFileName 自动完成文件名
SideBarTools 右键菜单工具
SublimeCodeIntel 代码自动补全
Terminal 快速打开终端工具 【修改 Packages\ Terminal\ Default.sublime-keymap
{ "keys": ["ctrl+shift+t"], "command": "open_terminal" }, 】
{ "keys": ["ctrl+t"], "command": "open_terminal_project_folder" }
VueSyntaxHighlighter .vue 文件代码高亮
ConvertToUTF8 内部包含多种编码（需手动调整打开文件的编码格式）
GitConflictResolver
TypeScriptSyntax

界面

File --Reopen with Encoding 用新编码格式打开文件
-- Reload with Encoding 用新编码格式加载文件，不会保存编码格式【右下即为文件编码】
-- Set File Encoding to 修改文件新编码格式（不会保存）
-- Save With Encoding—>UTF-8 with BOM 用新编码格式保存文件
View – syntax – Open all with current extension as.. 将此后缀文件一直识别为 xx 语法
- layout 分屏显示
Preferences --> Setting {"update_check":false, 不显示更新提示
"show_encoding": true,
"show_line_endings": true 在底部栏显示文件编码}
Tools --> Developer --> New Snippet... 添加代码块 //EX: ${1:xxx} 代码块可编辑内容
<scope>text.html,source.js</scope> 设置多个文件范围
Build System --> New Build System 新增编译系统

私人定制

Preferences.sublime-settings >> 个人全局配置

```
{
    "font_size": 8,
    "update_check": false,
    "remember_open_files": false,
    "auto_match_enabled":true,
    "auto_complete": true
}
```

Default（Windows）.sublime-keymap >> 快捷键

```
[
    {"keys": ["ctrl+e"], "command":"new_file"},
    {"keys": ["ctrl+q"], "command":"color_pick"},
    { "keys": ["ctrl+g"], "command": "cancel_build" }, 取消 build

  { "keys": ["ctrl+w"], "command": "find_next_conflict" },
  { "keys": ["ctrl+shift+e"], "command": "keep", "args": { "keep": "ours" } },
  { "keys": ["ctrl+shift+r"], "command": "keep", "args": { "keep": "theirs" } },
```

```
  { "keys": ["ctrl+shift+t"], "command": "keep", "args": { "keep": "ancestor" } },
  { "keys": ["ctrl+shift+d"], "command": "list_conflict_files" }
]
```

sdkPythonBuild.sublime-build >> 编译系统

```
{
    "shell_cmd": "start ${file}",          // 此为 windows 环境下。Linux 环境下应为 "shell_cmd": "${file}"，${file}为当前文件的绝对路径
    "selector": "text.html",
     "cancel": {"kill": true},                                  //关闭杀死终端
    "working_dir": "${project_path:${folder}}"            // echo $folder  项目根路径
}
```

快捷键

ctrl shift F 批量搜索
ctrl shift P 打开程序命令栏
ctrl shift → 选中匹配单词
ctrl alt ↑ 添加光标
ctrl D 选中多个相同的词

ctrl Q 选择颜色（修改）
ctrl E 新建文件（修改）
ctrl R 格式化代码（修改）
ctrl T 打开项目根目录（修改）

ctrl W 查找 git conflict
ctrl shift d 列出冲突文件
ctrl shift e 保留我的
ctrl shift r 保留对方的
ctrl shift t 保留之前的

编译系统

```
{
    // 此为 windows 环境下。Linux 环境下应为 "shell_cmd": "${file}",  ${file}为当前文件的绝对路径
    "shell_cmd": "start ${file}",
    "selector": "text.html",
}
```

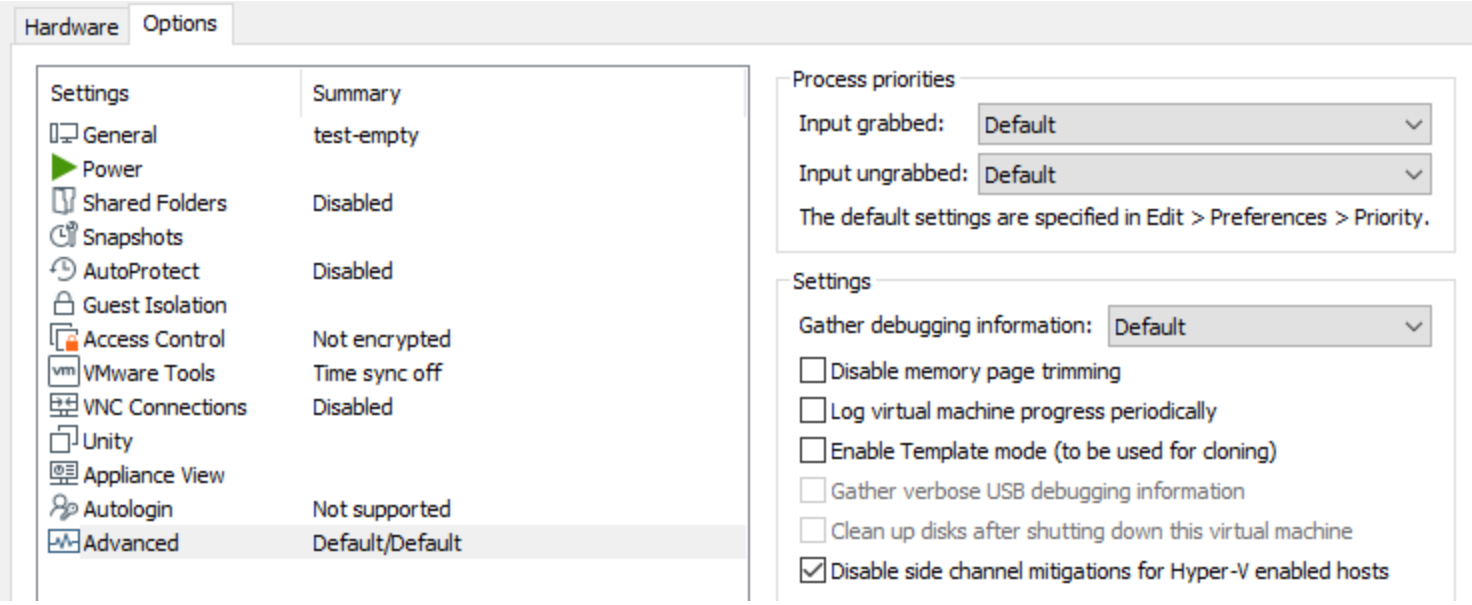
VMware

System Configuration

Enable Bios VT
Enable Hyper-v

Virutual Mechine Setting

side channel mitigation



Shortcut

ctrl + alt　鼠标退出虚拟机

Menu

File　　New Vitural Machine　-　Typical　-　installer disc image file(iso)

Edit - Virtual Network Editor

View - customize　　自定义显示窗口

VM - Settings - Hardware　　Network Adapter　　bridged

去除虚拟化　　WINHEX　Phoenix BIOS Editor　VMware SVGA 3D　　dxcpl 强制跳过 directx 检测

Network Configuration

Each virtual machine will be automatically assigned an IP.

概念：在虚拟机内配置后　网卡刷新会自动获取设置（ncpa.cpl 内配置则虚拟机的配置不能生效）

虚拟机网络设置网关均为 vmnetX 的 gateway address

配置网络 vmnetX　gateway addres 指的是此路由器的地址

Bridged（Vmnet0）　　虚拟机 <---> 虚拟机 <---> 主机　　能上网，需要网络，和主机在同一网段

虚拟机和主机--->Vmnet0--->网关

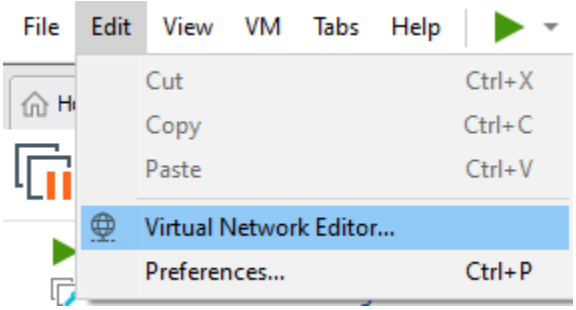
Host-only（Vmnet1）　　虚拟机 <---> 虚拟机 <----　主机　　不能上网，只能主机访问虚拟机

虚拟机---> Vmnet1--->主机--->Vmnet0--->网关

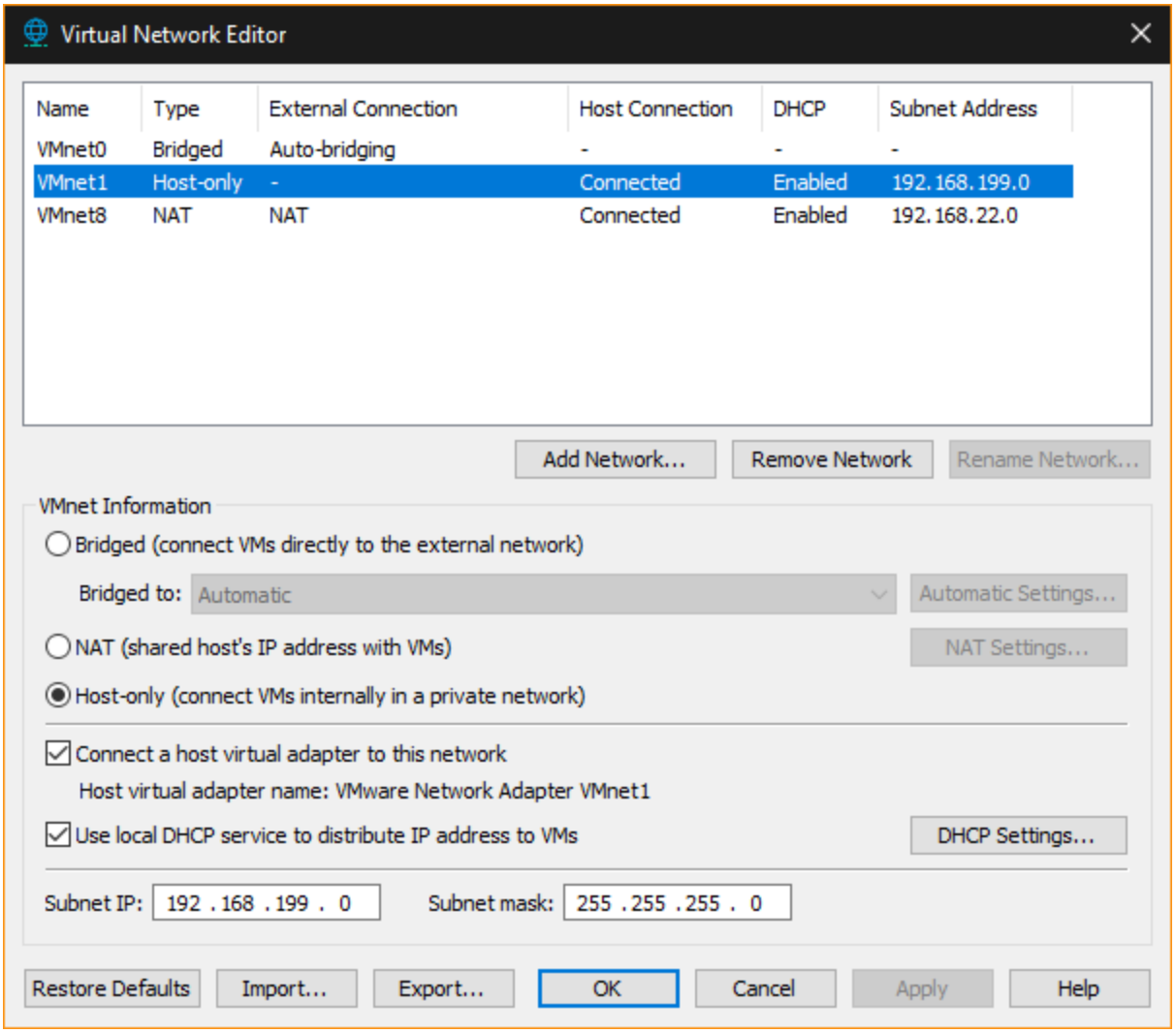
NAT　　（Vmnet8）　　虚拟机 <---> 虚拟机　　主机　　能上网，虚拟机和主机共享 ip，主机不能连接虚拟机

虚拟机---> Vmnet8--->主机--->Vmnet0--->网关

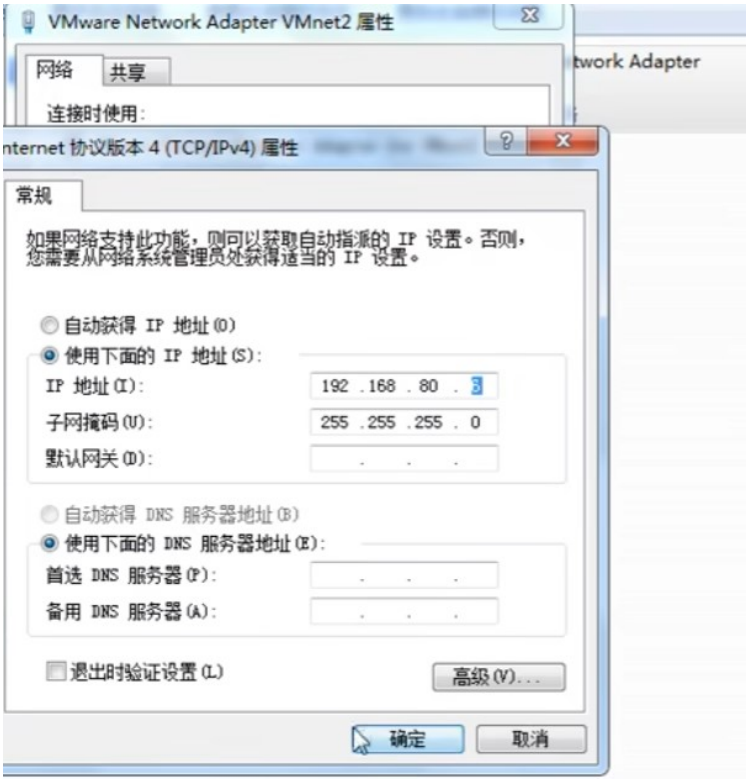
Custom　　虚拟机接到一个虚拟交换机 Vmnet 上



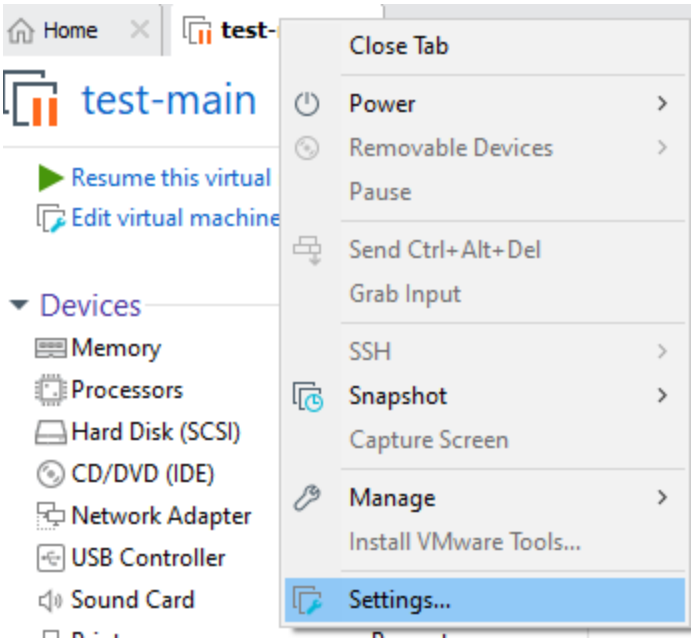
网卡配置，子网 ip 可修改网段（默认分配的网卡不能改网段）

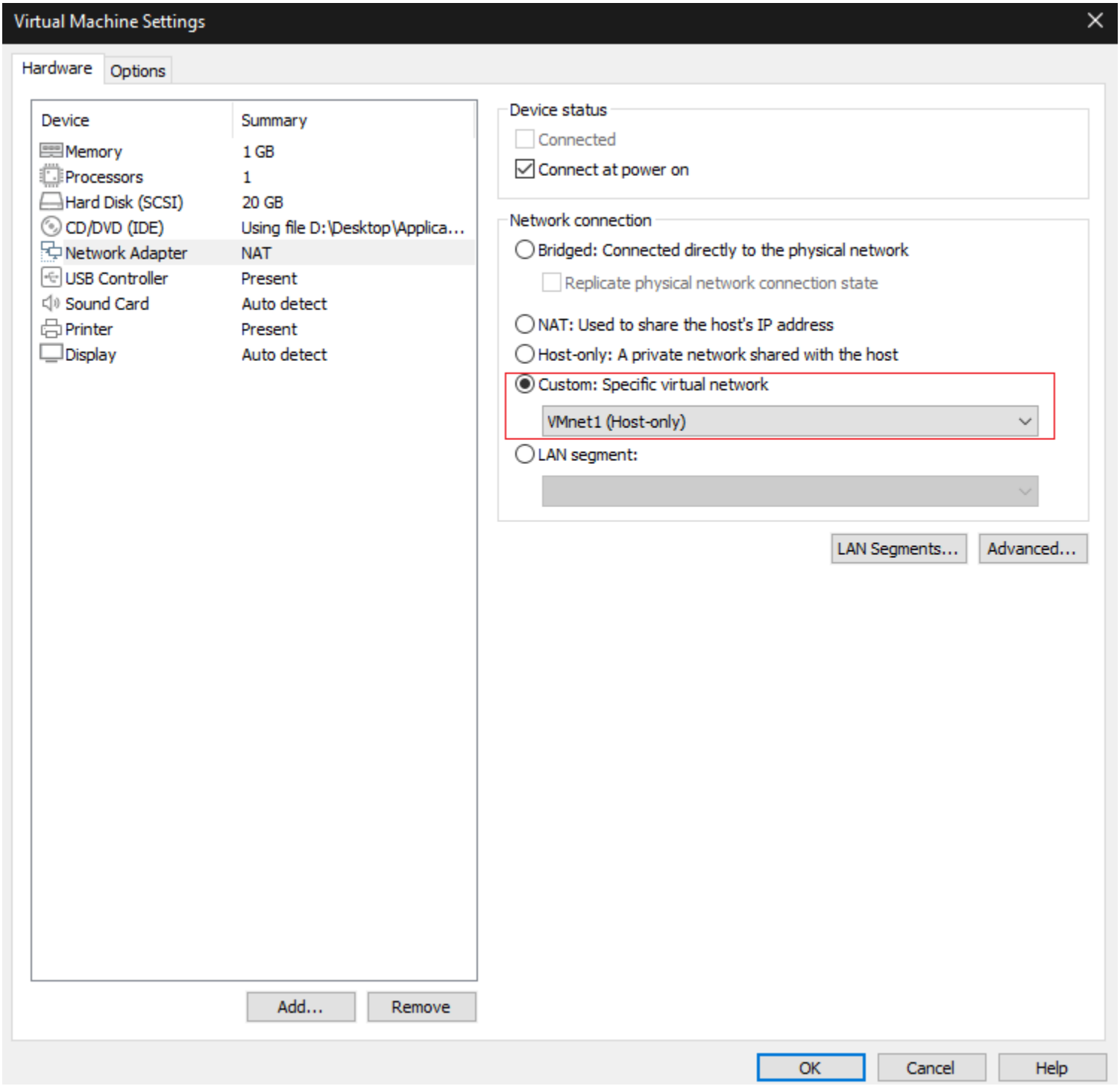


从主机固定虚拟网卡的 ip 地址

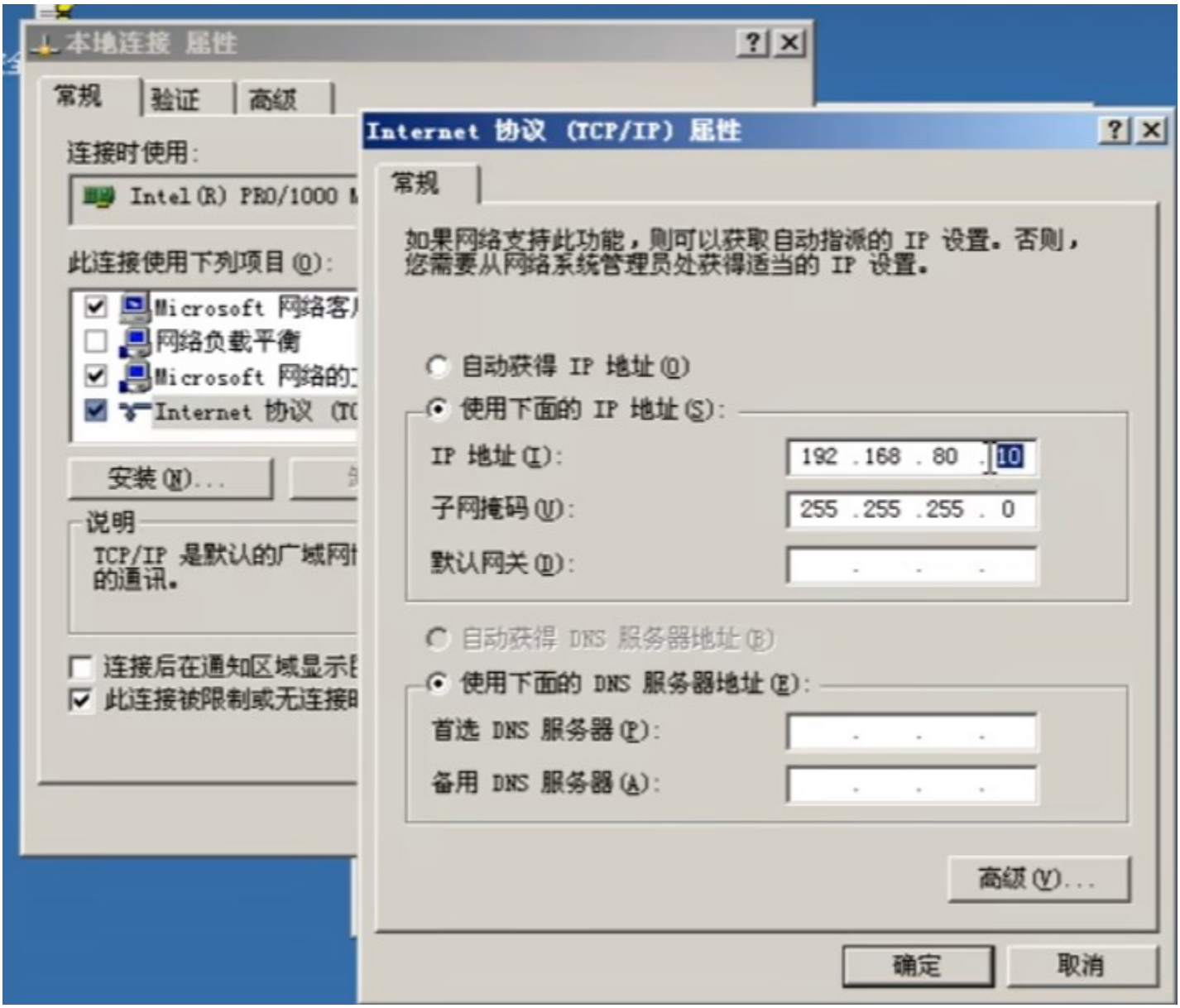


绑定虚拟机到虚拟网络中


Home
test-
test-main
Resume this virtual
Edit virtual machine
Devices
Memory
Processors
Hard Disk (SCSI)
CD/DVD (IDE)
Network Adapter
USB Controller
Sound Card
Close Tab
Power
Removable Devices
Pause
Send Ctrl+Alt+Del
Grab Input
SSH
Snapshot
Capture Screen
Manage
Install VMware Tools...
Settings...


Virtual Machine Settings
Hardware
Options
Device
Summary
Memory
1 GB
Processors
1
Hard Disk (SCSI)
20 GB
CD/DVD (IDE)
Using file D:\Desktop\Applica...
Network Adapter
NAT
USB Controller
Present
Sound Card
Auto detect
Printer
Present
Display
Auto detect
Add...
Remove
Device status
Connected
Connect at power on
Network connection
Bridged: Connected directly to the physical network
Replicate physical network connection state
NAT: Used to share the host's IP address
Host-only: A private network shared with the host
Custom: Specific virtual network
VMnet1 (Host-only)
LAN segment:
LAN Segments...
Advanced...
OK
Cancel
Help

固定操作系统网络 ip



linux Configuration

挂载共享目录：/mnt/hgfs（需要现在 VM setting 内开启共享文件夹）

cd /mnt/hgfs

vmware-hgfsclient 打印 VMShare 成功开启了共享功能

umount /mnt/hgfs && vmhgfs-fuse .host:/VMShare /mnt/hgfs 需要等一会儿才会挂载上（/etc/rc.d/rc.local 内配置可以自动启动挂载，chmod +x /etc/rc.d/rc.local 需要添加执行权限）

/etc/profile.d/ Add the command " umount /mnt/hgfs" and "vmhgfs-fuse .host:/VMShare /mnt/hgfs" into a bash file

vmware.sh

```
#!/bin/bash

# Unmount the directory if it's mounted
if mountpoint -q /mnt/hgfs; then
    umount /mnt/hgfs
fi
```

```
# Mount the VMware shared folder
vmhgfs-fuse .host:vmshare /mnt/hgfs
```

Other Configuration

BIOS Advanced installed O/S 使计算机能支持一些老版本系统（win95）
Boot +Hard Drive 硬盘启动 CD-ROM Drive 光驱启动 Removable Devices U 盘启动
Network boot from Intel E1000 网卡启动（启动远程服务器）

MinGW

Minimalist GNU for Windows （含头文件和端口库的 编译器）
gcc -v 查看版本
g++ a.cpp [–o xx.exe] 编译生成 exe 文件

eclipse

修改输出 class 路径 Java Build Path Default output folder(Allow output folders for source folders)
快速生成 get set 方法 source Generate Getters and Setters:
ERROR: some projects cannot be imported because they already exist in the workspace 修改.project 里工程名

VScode

主题 Nebula Theme（界面） vscode-icons（图标） prettier javascript console utils terminals manager typescript hero
快捷键 ctrl shift P 控制台 Terminal: Edit Configuration
ctrl P 查找文件
ctrl , 点击右上角打开配置文件（Editor：Format On Save Editor：Suggest Selection）
alt 多光标
ctrl alt 多行光标
F2 批量修改变量名
ctrl shift R 重构
alt shift f 格式化代码
（alt shift o） 格式化导入（typescript hero）
ctrl K ctrl 0 折叠所有代码
ctrl K ctrl J 展开所有代码

View > Terminal 打开 vs 控制台
Workbench > Tree: Render Indent Guides 去掉目录树的虚线

配置——

.vscode/settings.json >>

```
{
    "liveServer.settings.port": 5502,
    "typescript.validate.enable": false,
    "javascript.validate.enable": false,
}
```

.vscode/launch.json >>
other >>

```
  "editor.suggestSelection": "first",
  "editor.fontSize": 14,                    字体大小
  "editor.formatOnSave": true,              自动保存
  "prettier.singleQuote": false,            使用单引号替换双引号
  "prettier.trailingComma": "none",         对象尾逗号是否显示
```

```
  "prettier.arrowParens":"avoid"          箭头函数是否省略括号
  “files.associations”:{ "*.scss":"scss" }      解决文件关联报错问题
"workbench.colorCustomizations": {
  "editorUnnecessaryCode.border": "#0000ff"
}
```

Terminal Manager >> ctrl shift p Configure Configuration 配置启动项

```
{
  "autorun": true,
  "terminals": [
    {
      "name": "shop-ope",
      "focus": true,
      "commands": [
        "cd shopping-mall-operation-web",
        "npm run start"
      ]
    },
    {
      "name": "shop-commer",
      "focus": false,
      "commands": [
        "cd shopping-mall-commercial-tenant-web",
        "npm run dev"
      ]
    },
    {
      "name": "shop-c",
      "focus": false,
      "commands": [
        "cd shopping-mall-c-web",
        "npm run dev invoice"
      ]
    }
  ]
}
```

ERROR Vetur can't findtsconfig,json'or jsconfig.json' vetur: Ignore Project Warning

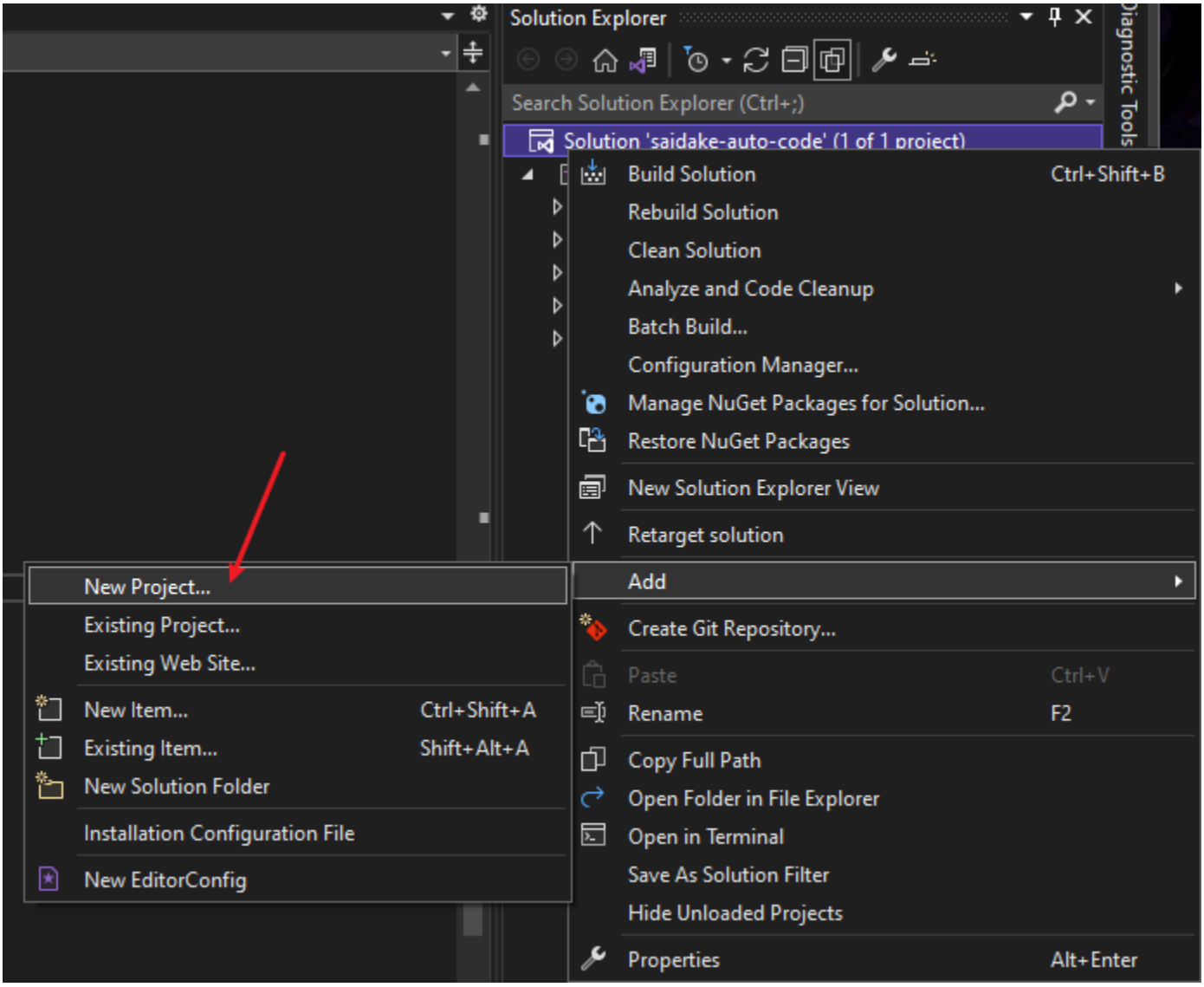
修改环境变量配置后需要重启 vscode

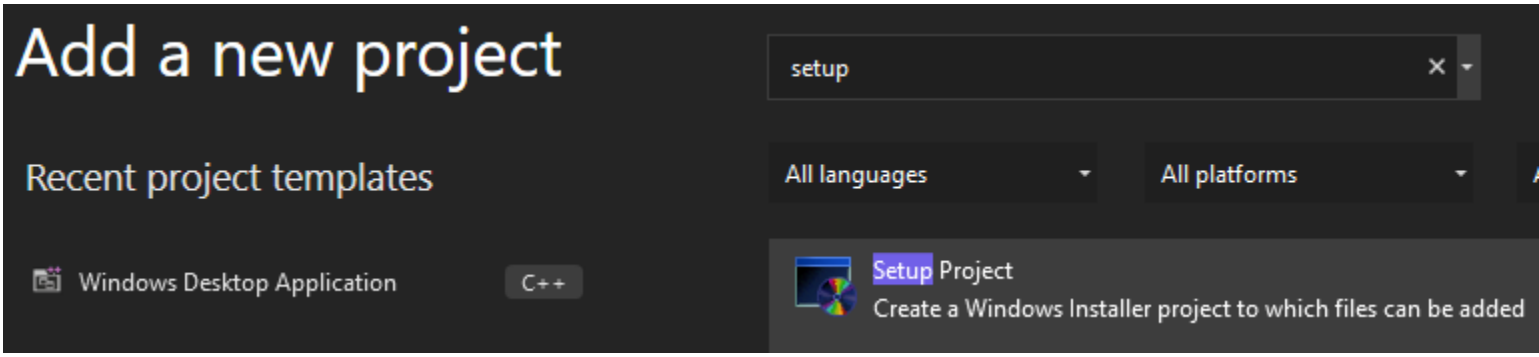
preview 取消预览

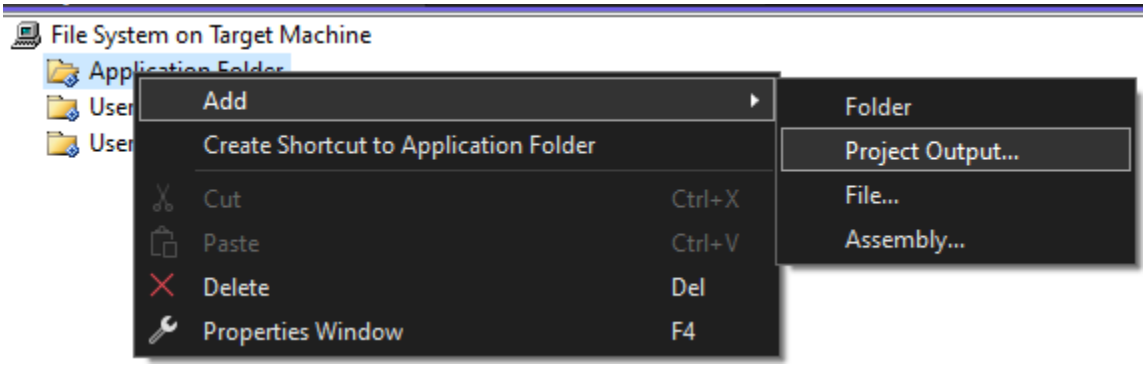
visual studio

插件：install project

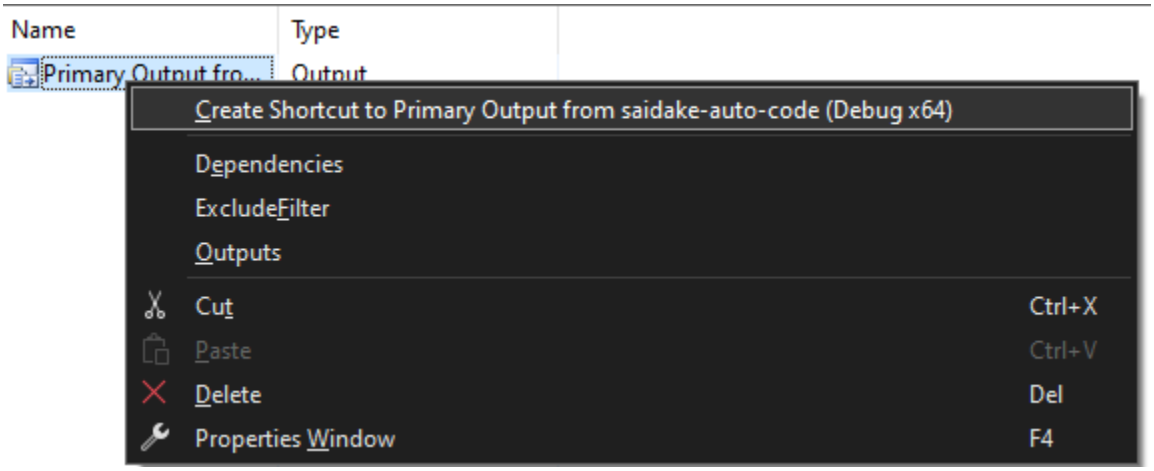
打包项目：

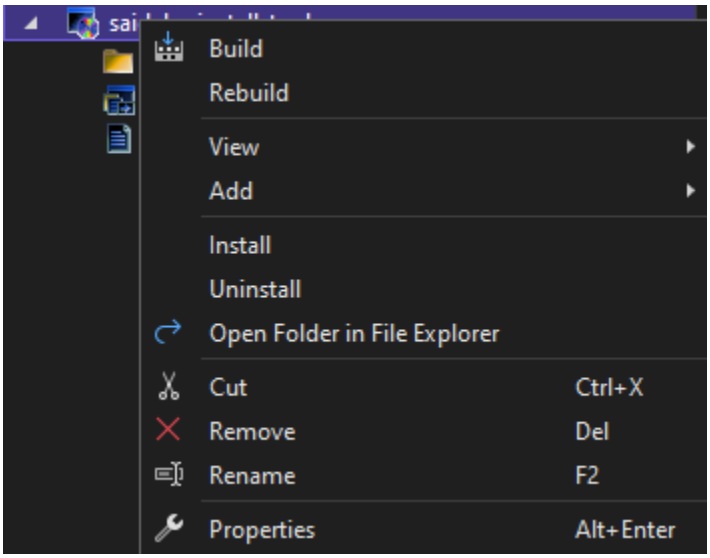

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'saidake-auto-code' (1 of 1 project)
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Analyze and Code Cleanup
Batch Build...
Configuration Manager...
Manage NuGet Packages for Solution...
Restore NuGet Packages
New Solution Explorer View
Retarget solution
Add
Create Git Repository...
Paste Ctrl+V
Rename F2
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Save As Solution Filter
Hide Unloaded Projects
Properties Alt+Enter
New Project...
Existing Project...
Existing Web Site...
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Solution Folder
Installation Configuration File
New EditorConfig

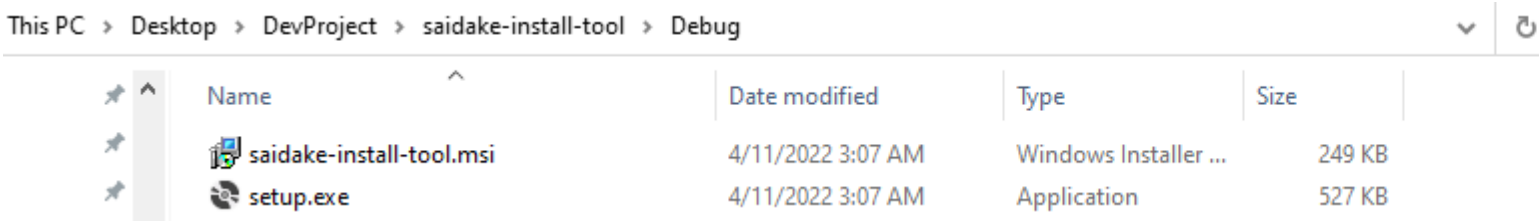

Add a new project
setup
Recent project templates
All languages
All platforms
Windows Desktop Application
C++
Setup Project
Create a Windows Installer project to which files can be added


File System on Target Machine
Add
Create Shortcut to Application Folder
Cut Ctrl+X
Paste Ctrl+V
Delete Del
Properties Window F4
Folder
Project Output...
File...
Assembly...

创建快捷方式，并挪到 User's Desktop







GoLand

GOPATH　设置为项目目录

UltraSO

9.7.1.3519　　王涛 7C81-1689-4046-626F

打开文件

启动　写入硬盘镜像（写入方式选择为“USB-HDD+”）

FlashFXP

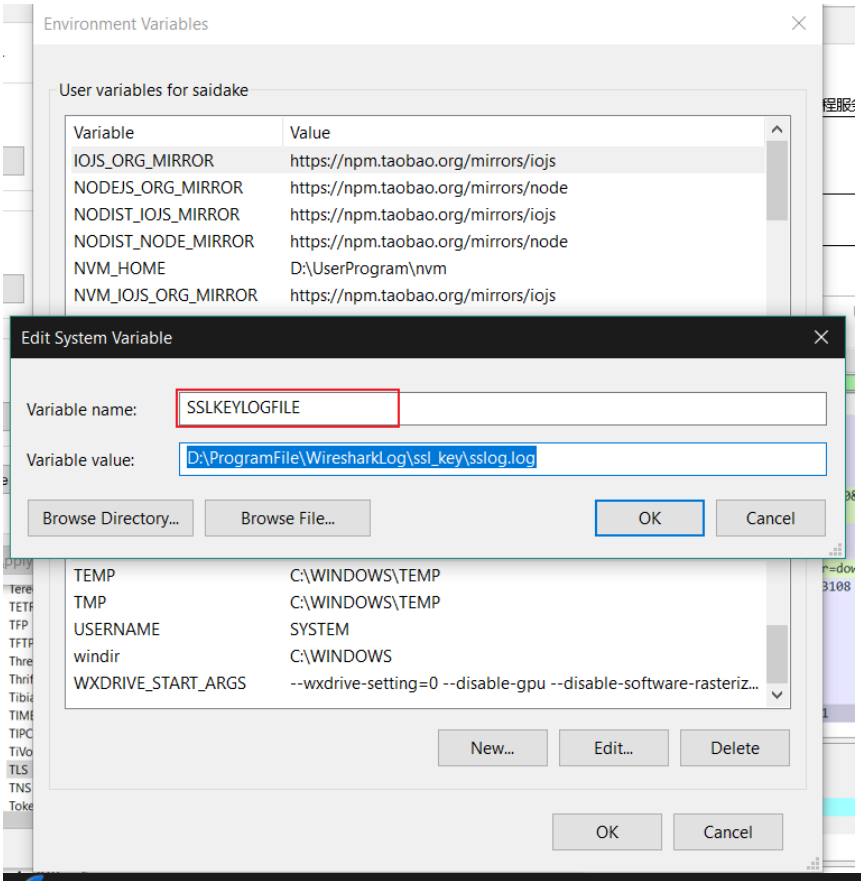
FTP 账号连接　21 端口

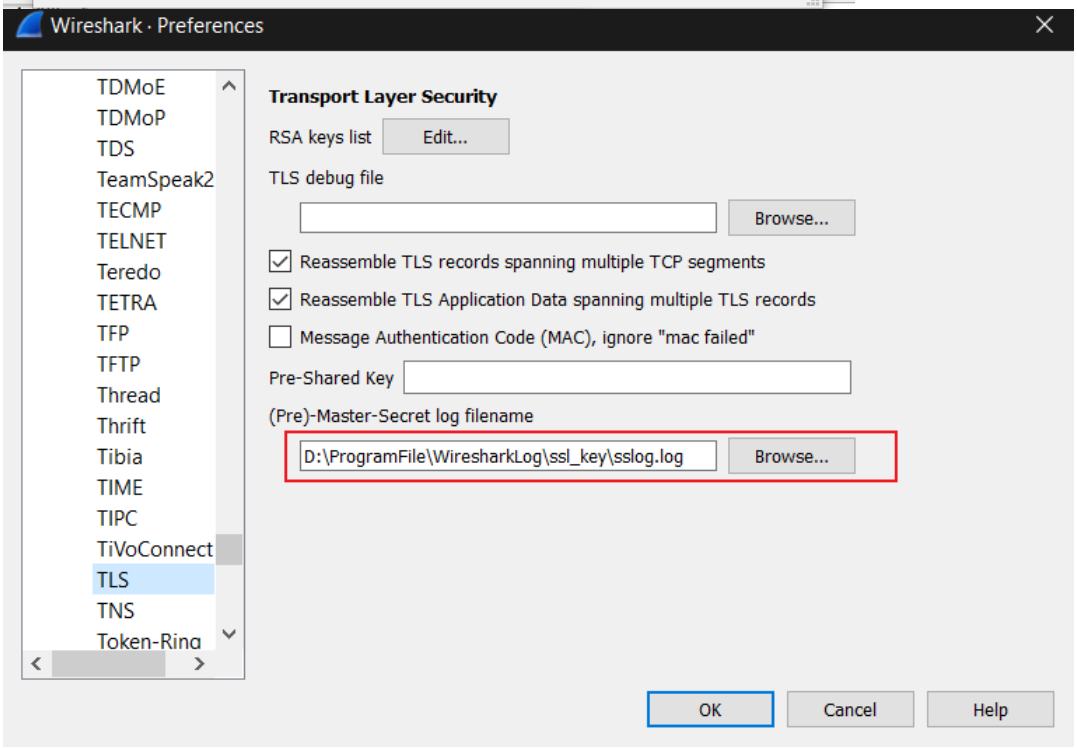
Sites　-Site Manager　网站管理

wireShark

安装配置

1. 配置 https 抓包





捕捉 cookie >>

File --> Export Specified Packets 导出特定数据包

Perferences --> Name Resolution --> Resolve network(IP) addresses 显示域名等解析信息

User Interface --> Layout 修改界面布局

View --> Time Display Format 显示时间

Statistics --> Show address resolution 显示 ip 域名解析结果

Analyze --> Expert Info 专家信息

右键--> Manually Resolve Address 手动解析 ip 地址

Mark Packet 标记包

Packet Comment 添加包注释

Follow --> TCP Stream

快捷栏红色方块 --> 修改默认根据协议的颜色配置方案

介绍

广播域：站点发出一个广播信号后能接收到这个信号的范围。通常来说一个局域网就是一个广播域。

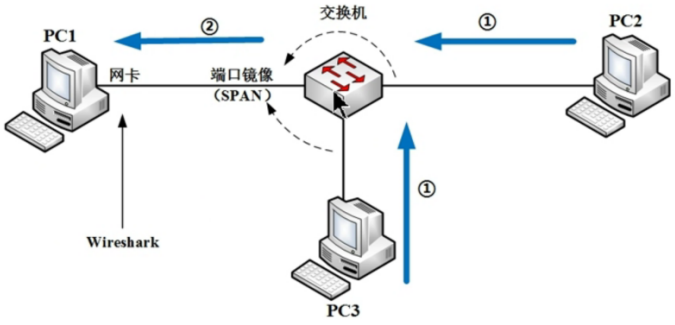
冲突域：指一个站点向另一个站点发出信号。除目的站点外，其他也能接受到信号的站点就构成一个冲突域

混杂模式：指一台机器能够接收所有经过它的数据流，而不论其目的地址是否是他。

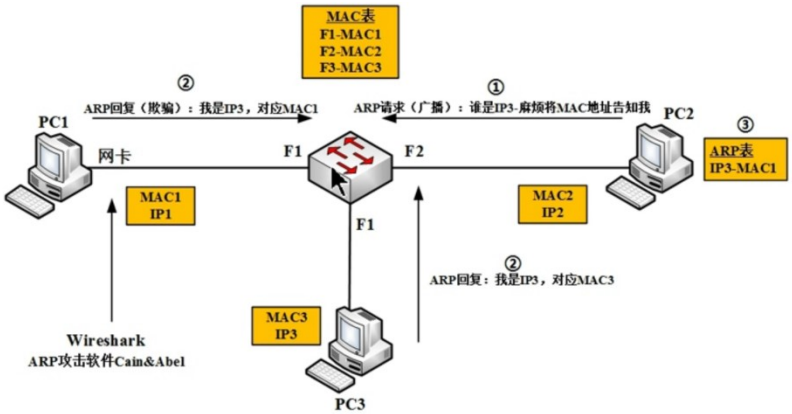
本机环境： 直接抓本机网卡进出的流量

集线器环境： 流量防洪，同一冲突域，被淘汰 【物理层】

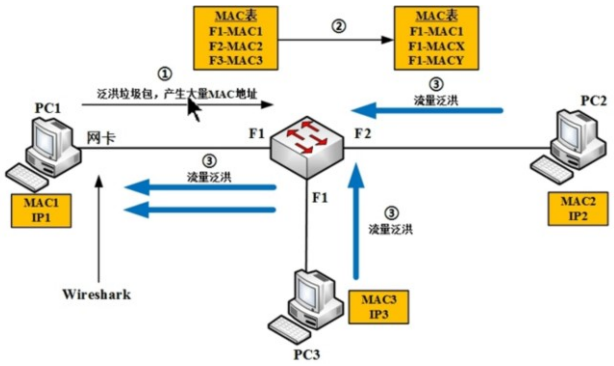
交换机环境： 端口镜像 SPAN（将交换机端口的流量拷贝到本机网卡来，本机网卡设置为混杂模式）【数据链路层】



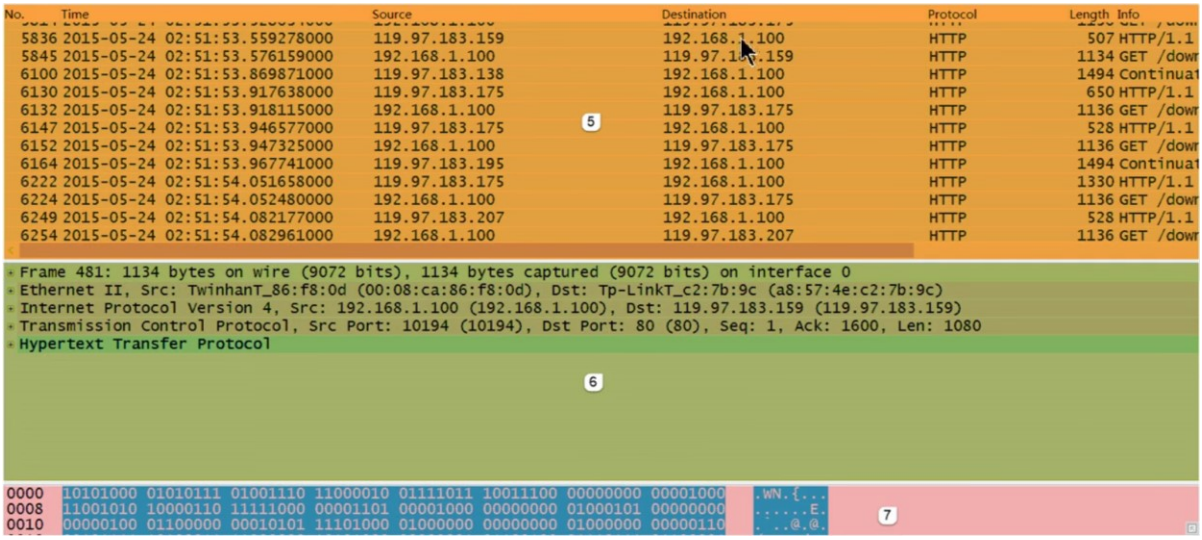
ARP 欺骗 Cain&Abel 局域网流量劫持（原本本机会丢弃包，但是做了欺骗性回复，抓到 IP3 的包）



MAC 泛洪（在交换机中挤出其他电脑的 MAC 地址，这时其他电脑在交换机上查找不到 MAC 地址，流量无法流出，会对外泛洪）



使用



数据包列表区：每一条都是一个数据包
数据包信息区：　Tp-LinkT_c2:7b:9c（a8:57:4e:c2:7b:9c）　MAC 地址 头三位 显示设备厂商
数据包字节区：　16 进制 查看包长度

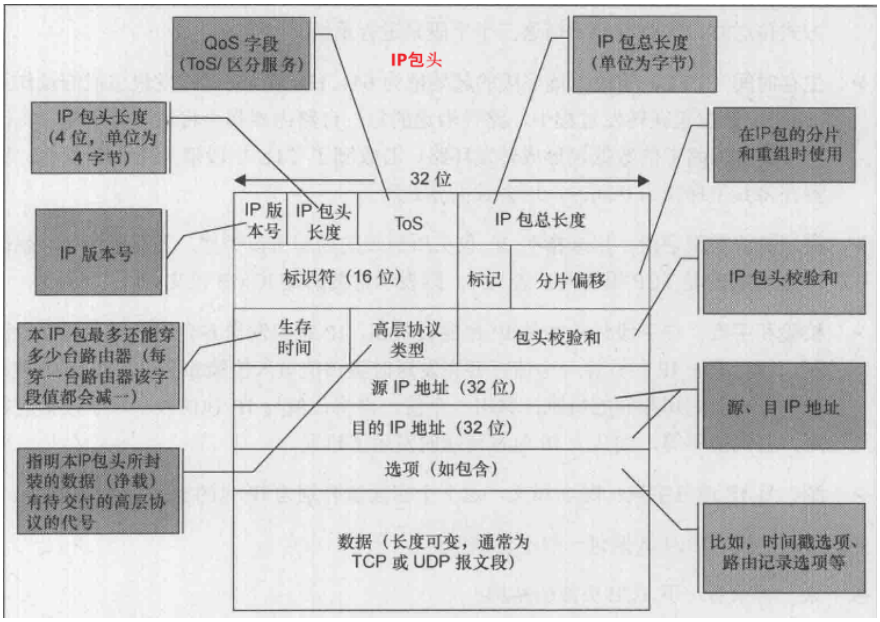
抓包过滤器：　协议 Proto：　ether　ip　tcp　udp　http　ftp
　　方向 Dir：　src　dst　　（源，目的地）
　　类型 Type：　host　net　port　　（主机，网段，端口）
　　运算符：　&&　||　!
　过滤地址和协议：　src host 192.168.1.1 && dst port 80　　抓取源地址为 192.168.1.1，目的为 80 端口的流量
　　host 192.168.1.1 || host 192.168.1.1　　抓取 192.168.1.1 和 192.168.1.2 的流量
　　! broadcast　不抓取广播包

　过滤 MAC 地址：ether host 00:88:ca:86:f8:0d　　局域网过滤 mac 地址
　　ether src host 00:88:ca:86:f8:0d　局域网这个 mac 地址发出的包
　过滤端口　　port 80　!port 80　dst port 80　tcp.port == 80
　过滤协议　　arp　icmp　http
　综合过滤　　host 192.168.1.100 && port 8080
显示过滤器：运算符：　==　!=　>　<　>=　<=
　　逻辑操作符：　eq　and　or　xor　not　　xor 有且只有一个条件被满足
　　ip 地址：　ip.addr　ip.src　ip.dst
　　网址过滤：　http.host contains baidu.com 是模糊匹配
　　端口过滤：　tcp.port　tcp.srcport　tcp.dstport　tcp.flag.syn　tcp.flag.ack　// tcp.flag.ack == 80　ip.src

== 192.168.1.100 and tcp.sdtport ==80

协议： arp tcp ucp not http hot arp

报文分析——



Frame 31: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{CCBB5F0B-12A3-449E-A9C1-C81A3AD26C23}, id 0 物理层（数据帧概况）

31 号帧，线路 54 字节，实际捕获 54 字节

Interface id: 0 (\Device\NPF_{CCBB5F0B-12A3-449E-A9C1-C81A3AD26C23}) #接口 id

Interface name: \Device\NPF_{CCBB5F0B-12A3-449E-A9C1-C81A3AD26C23}

Interface description: WLAN

Encapsulation type: Ethernet (1)

Arrival Time: Nov 25, 2021 00:39:27.428212000 China Standard Time #捕获日期和时间

[Time shift for this packet: 0.000000000 seconds]

Epoch Time: 1637771967.428212000 seconds

[Time delta from previous captured frame: 0.000121000 seconds] #此包与前一包的时间间隔

[Time delta from previous displayed frame: 0.000121000 seconds]

[Time since reference or first frame: 0.218009000 seconds] #此包与第一帧的时间间隔

Frame Number: 31 #帧序号

Frame Length: 54 bytes (432 bits) #帧长度

Capture Length: 54 bytes (432 bits) #捕获长度

[Frame is marked: False] #此帧是否做了标记

[Frame is ignored: False] #此帧是否被忽略

[Protocols in frame: eth:ethertype:ip:tcp] #帧内封装的协议层次结构

[Coloring Rule Name: TCP SYN/FIN] #着色标记的协议名称

[Coloring Rule String: tcp.flags & 0x02 || tcp.flags.fin == 1] #着色规则显示的字符串

Ethernet II, Src: Shenzhen_b3:2b:39 (4c:77:66:b3:2b:39), Dst: BeijingX_ee:b9:6d (9c:9d:7e:ee:b9:6d) 数据链路层（以太网帧头部信息）

Destination: BeijingX_ee:b9:6d (9c:9d:7e:ee:b9:6d) #目标 MAC 地址

Source: Shenzhen_b3:2b:39 (4c:77:66:b3:2b:39) #源 MAC 地址

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 192.168.31.182, Dst: 124.14.12.167 网络层

0100 .... = Version: 4 #互联网协议 IPv4

.... 0101 = Header Length: 20 bytes (5) #IP 包头部长度

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 40 #IP 包总长度

Identification: 0xd98b (55691) #标志字段

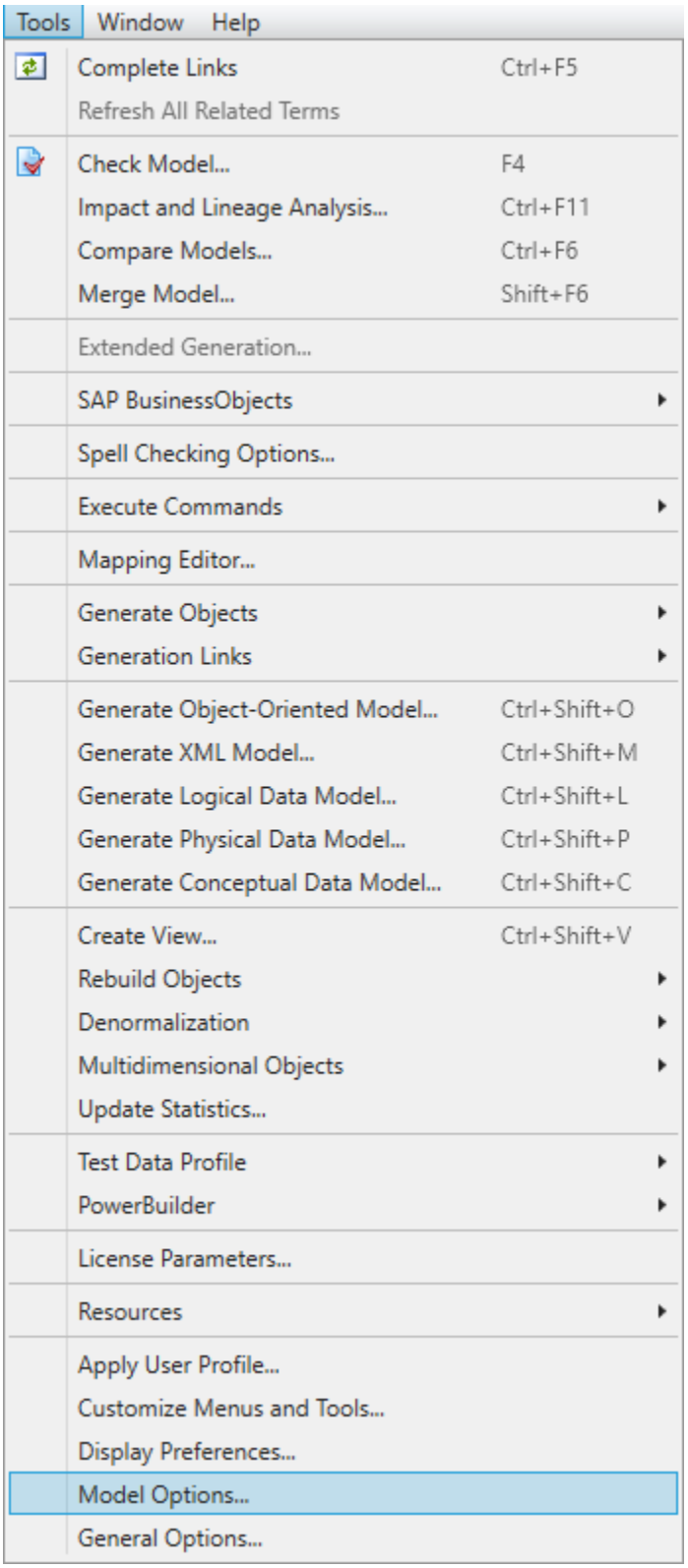
Flags: 0x40, Don't fragment #标记字段

...0 0000 0000 0000 = Fragment Offset: 0 #分的偏移量

Time to Live: 128 #生存期 TTL
Protocol: TCP (6) #此包内封装的上层协议为 TCP
Header Checksum: 0xb830 [validation disabled] #头部数据校验
[Header checksum status: Unverified]
Source Address: 192.168.31.182 #源 IP 地址
Destination Address: 124.14.12.167 #目标 IP 地址
Transmission Control Protocol, Src Port: 53935, Dst Port: 443, Seq: 1, Ack: 40, Len: 0 传输层
Source Port: 53935 #源端口号
Destination Port: 443 #目标端口号
[Stream index: 3] #tcp 分段，需要跟踪流 follow tcp stream
[Conversation completeness: Incomplete (28)]
[TCP Segment Len: 0]
Sequence Number: 1 (relative sequence number) #相对序列号
Sequence Number (raw): 3427301706
[Next Sequence Number: 2 (relative sequence number)] #下一个序列号
Acknowledgment Number: 40 (relative ack number) #确认序列号
Acknowledgment number (raw): 3147717129
0101 .... = Header Length: 20 bytes (5)
Flags: 0x011 (FIN, ACK) #TCP 标记字段
Window: 8279 #流量控制的窗口大小
[Calculated window size: 8279]
[Window size scaling factor: -1 (unknown)]
Checksum: 0xf6c2 [unverified] #TCP 数据段校验
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]
Hypertext Transfer Protocol （会话层，表示层-->）应用层
[truncated]GET /kvcollect?BossId=3647&Pwd=1005892234
Host: btrace.qq.com\r\n
Accept: */*Content-Type: application/x-www-form-urlencoded\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 5.00; Windows 98)\r\n
Pragma: no-cache\r\n
\r\n
[Full request URI [truncated]: http://btrace.qq.com/kvcollect?BossId=3647&Pwd=1005892234]
[HTTP request 1/1]
[Response in frame: 5239]

powerdesigner

初始化


Tools
Window
Help
Complete Links Ctrl+F5
Refresh All Related Terms
Check Model... F4
Impact and Lineage Analysis... Ctrl+F11
Compare Models... Ctrl+F6
Merge Model... Shift+F6
Extended Generation...
SAP BusinessObjects
Spell Checking Options...
Execute Commands
Mapping Editor...
Generate Objects
Generation Links
Generate Object-Oriented Model... Ctrl+Shift+O
Generate XML Model... Ctrl+Shift+M
Generate Logical Data Model... Ctrl+Shift+L
Generate Physical Data Model... Ctrl+Shift+P
Generate Conceptual Data Model... Ctrl+Shift+C
Create View... Ctrl+Shift+V
Rebuild Objects
Denormalization
Multidimensional Objects
Update Statistics...
Test Data Profile
PowerBuilder
License Parameters...
Resources
Apply User Profile...
Customize Menus and Tools...
Display Preferences...
Model Options...
General Options...

显示注释

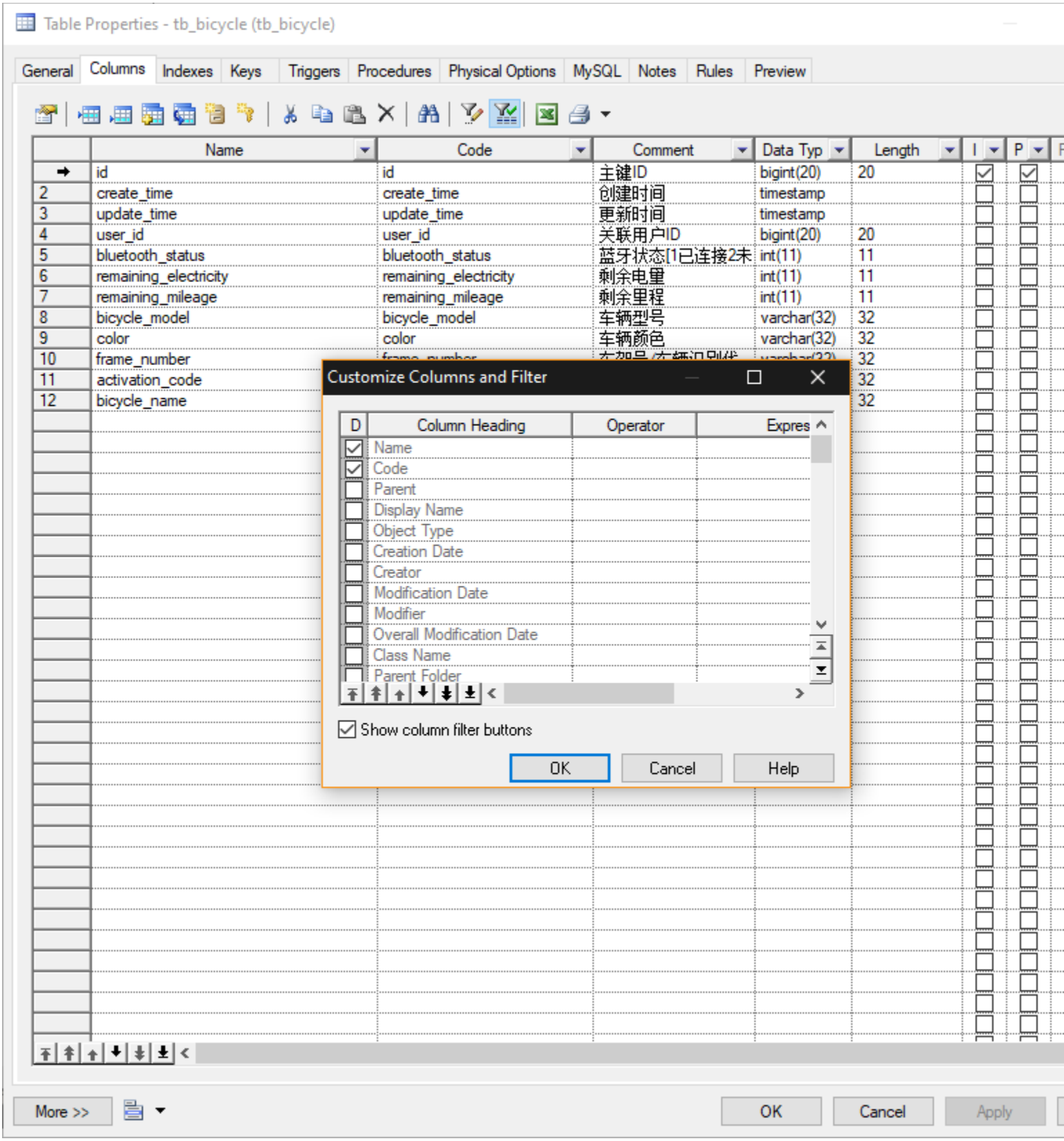
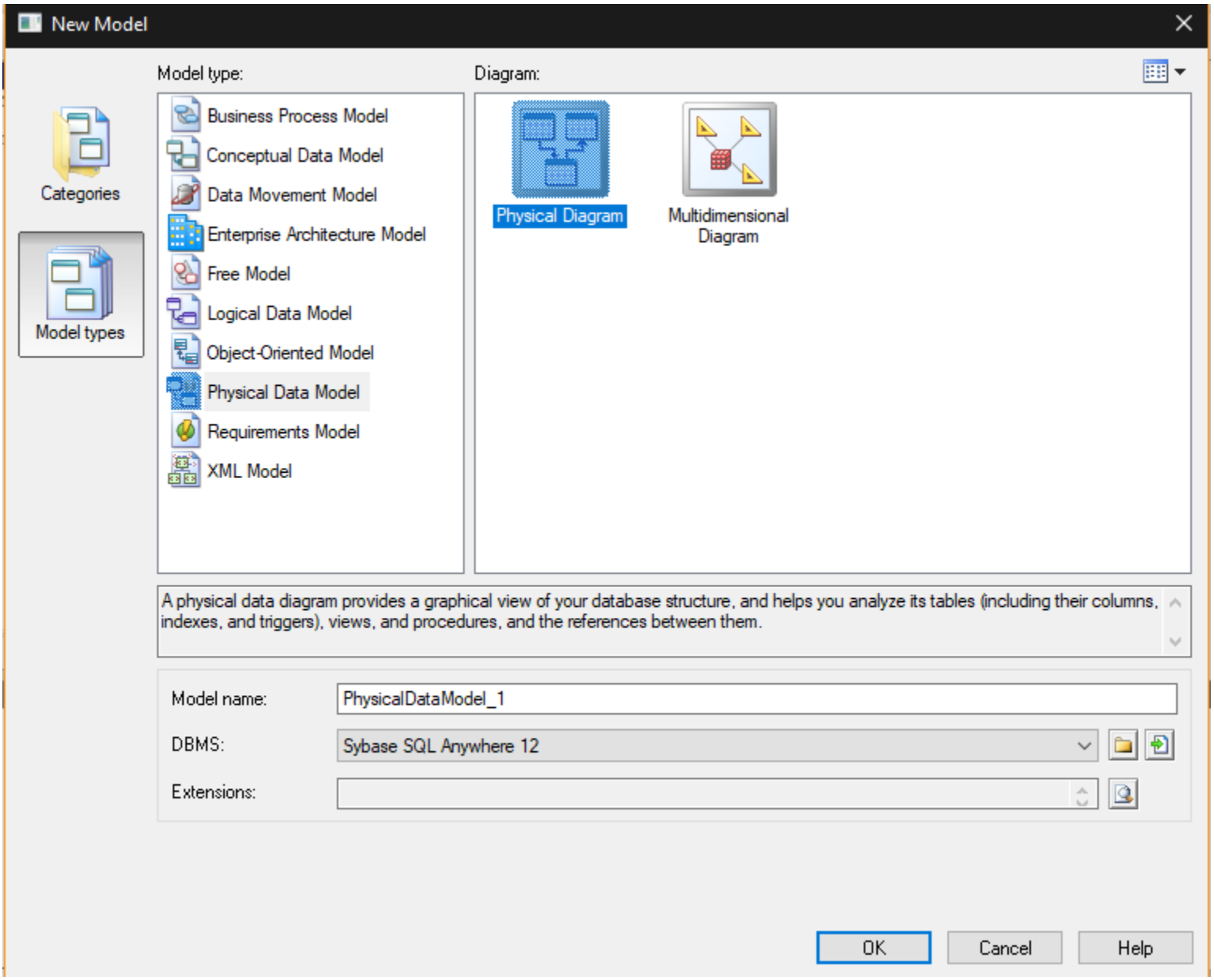
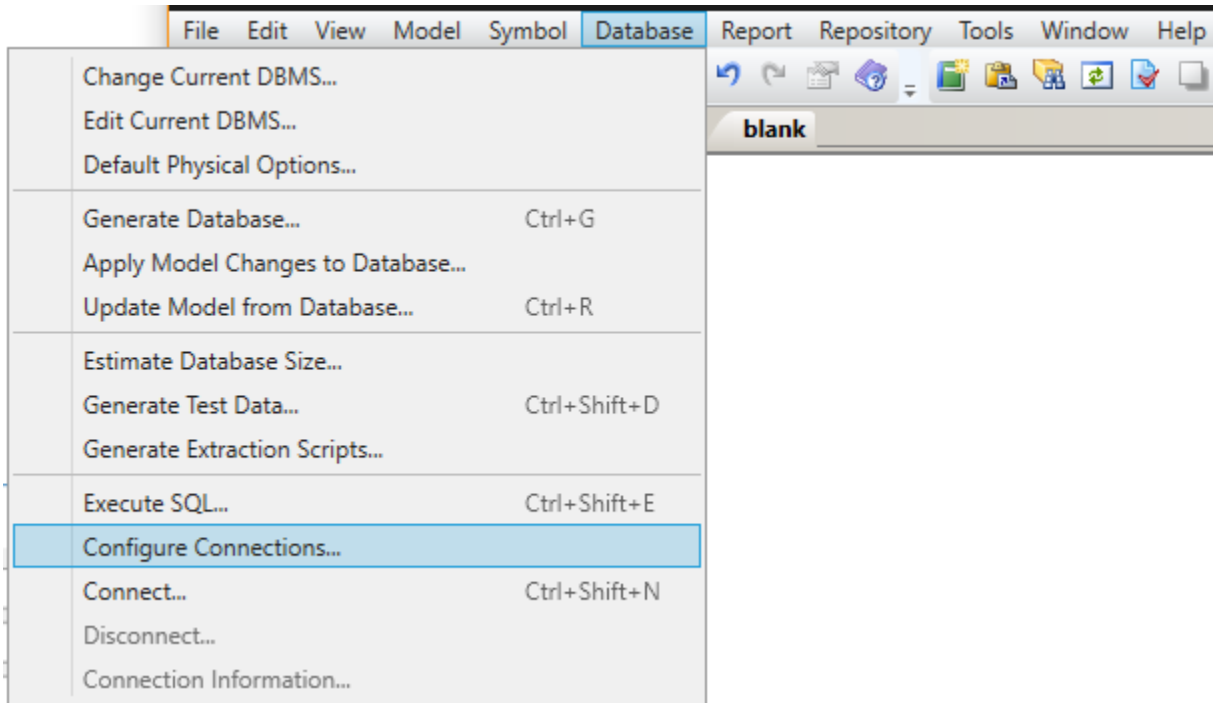


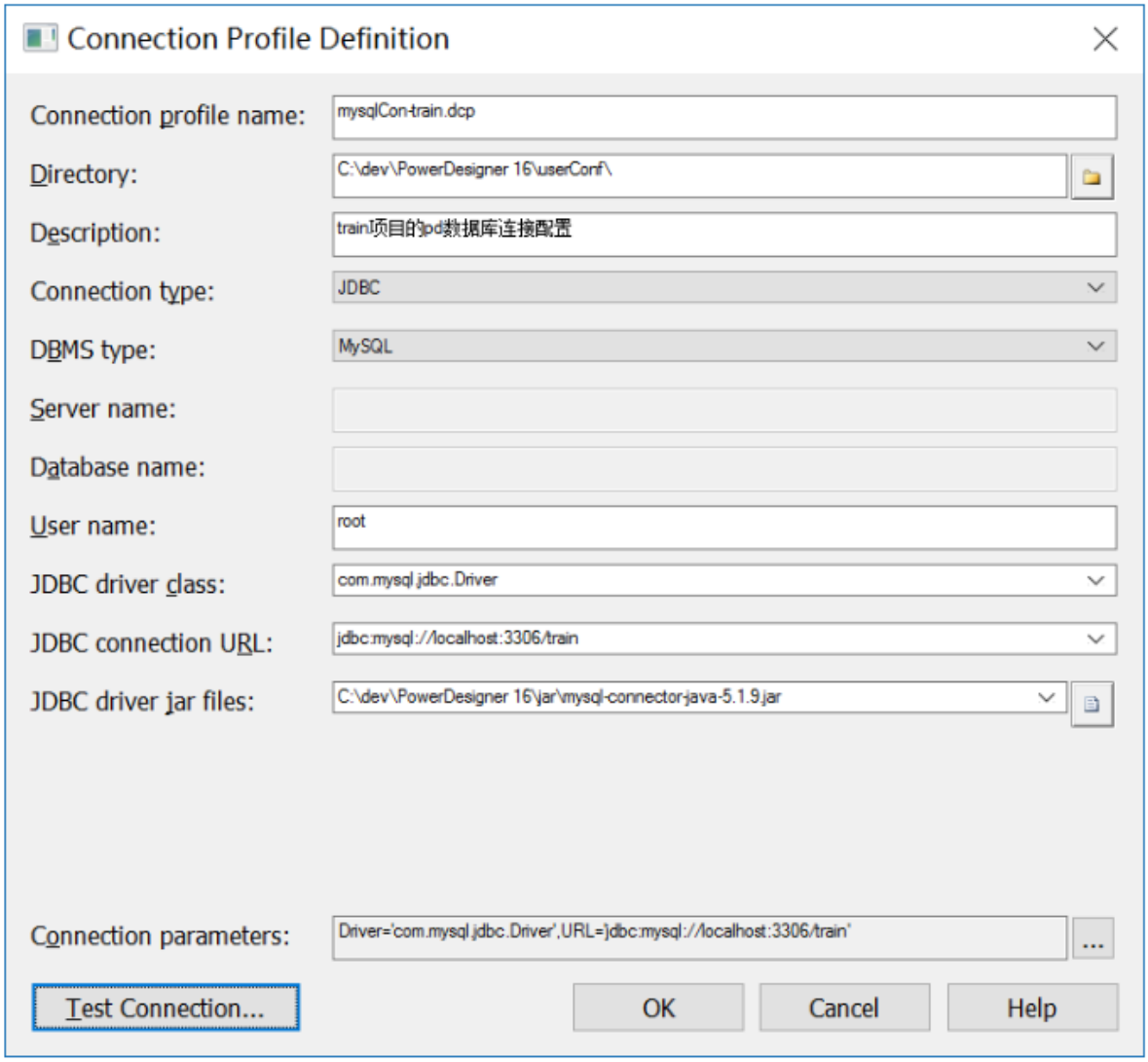
Table Properties - tb_bicycle (tb_bicycle)
General
Columns
Indexes
Keys
Triggers
Procedures
Physical Options
MySQL
Notes
Rules
Preview
Name
Code
Comment
Data Typ
Length
I
P
1 id id 主键ID bigint(20) 20
2 create_time create_time 创建时间 timestamp
3 update_time update_time 更新时间 timestamp
4 user_id user_id 关联用户ID bigint(20) 20
5 bluetooth_status bluetooth_status 蓝牙状态[1已连接2未 int(11) 11
6 remaining_electricity remaining_electricity 剩余电量 int(11) 11
7 remaining_mileage remaining_mileage 剩余里程 int(11) 11
8 bicycle_model bicycle_model 车辆型号 varchar(32) 32
9 color color 车辆颜色 varchar(32) 32
10 frame_number 32
11 activation_code 32
12 bicycle_name 32
Customize Columns and Filter
D
Column Heading
Operator
Expres
Name
Code
Parent
Display Name
Object Type
Creation Date
Creator
Modification Date
Modifier
Overall Modification Date
Class Name
Parent Folder
Show column filter buttons
OK
Cancel
Help
More >>
OK
Cancel
Apply

创建 model 项目


New Model
Model type:
Diagram:
Categories
Model types
Business Process Model
Conceptual Data Model
Data Movement Model
Enterprise Architecture Model
Free Model
Logical Data Model
Object-Oriented Model
Physical Data Model
Requirements Model
XML Model
Physical Diagram
Multidimensional Diagram
A physical data diagram provides a graphical view of your database structure, and helps you analyze its tables (including their columns, indexes, and triggers), views, and procedures, and the references between them.
Model name:
PhysicalDataModel_1
DBMS:
Sybase SQL Anywhere 12
Extensions:
OK
Cancel
Help

导出 sql

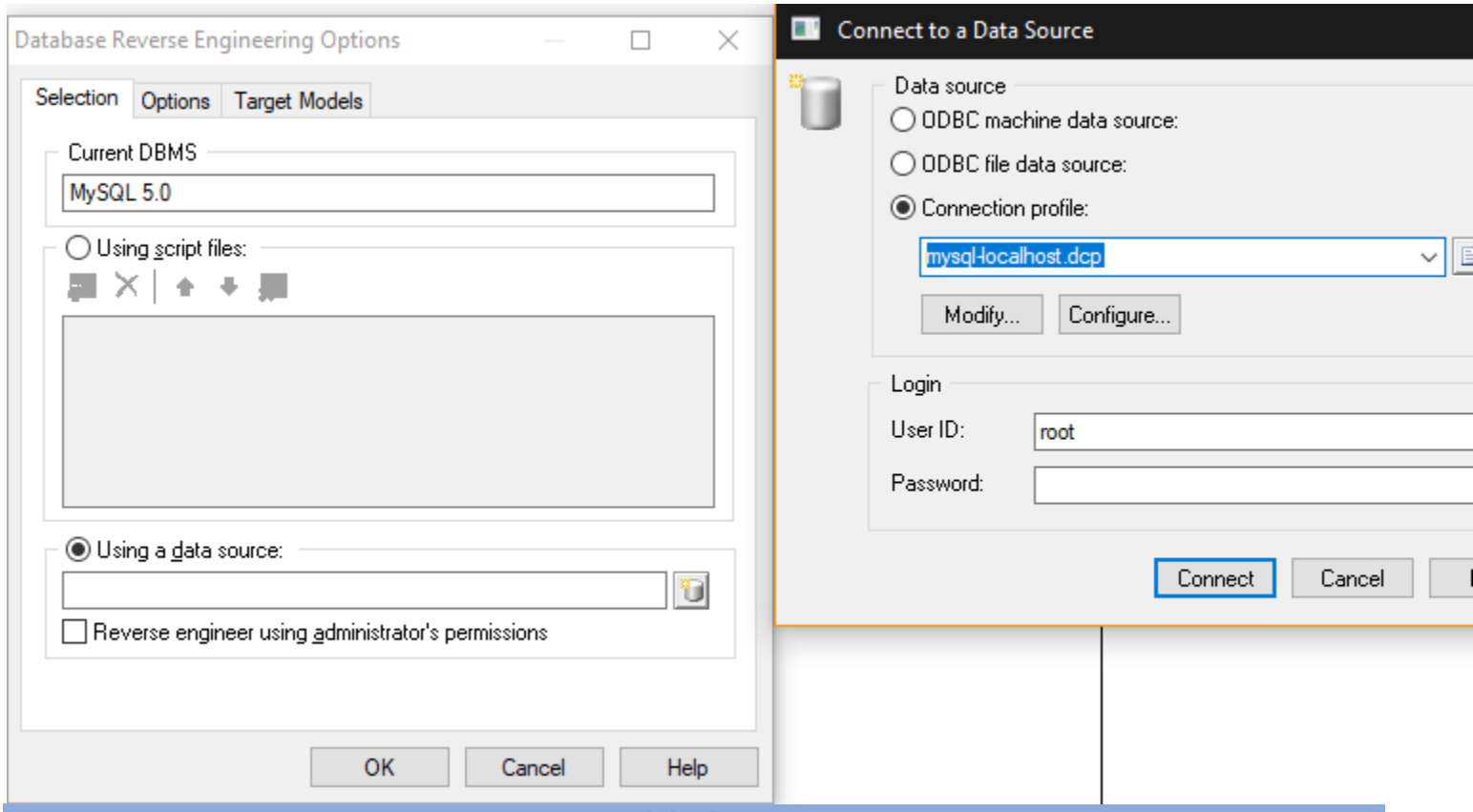




Set JAVA_HOME=C:\Program Files (x86)\Java\jdk1.8.0_31

Set CLASSPATH = C:\Program Files (x86)\Sybase\PowerDesigner 16\mysql-connector-java-5.1.25.jar

pdshell16.exe



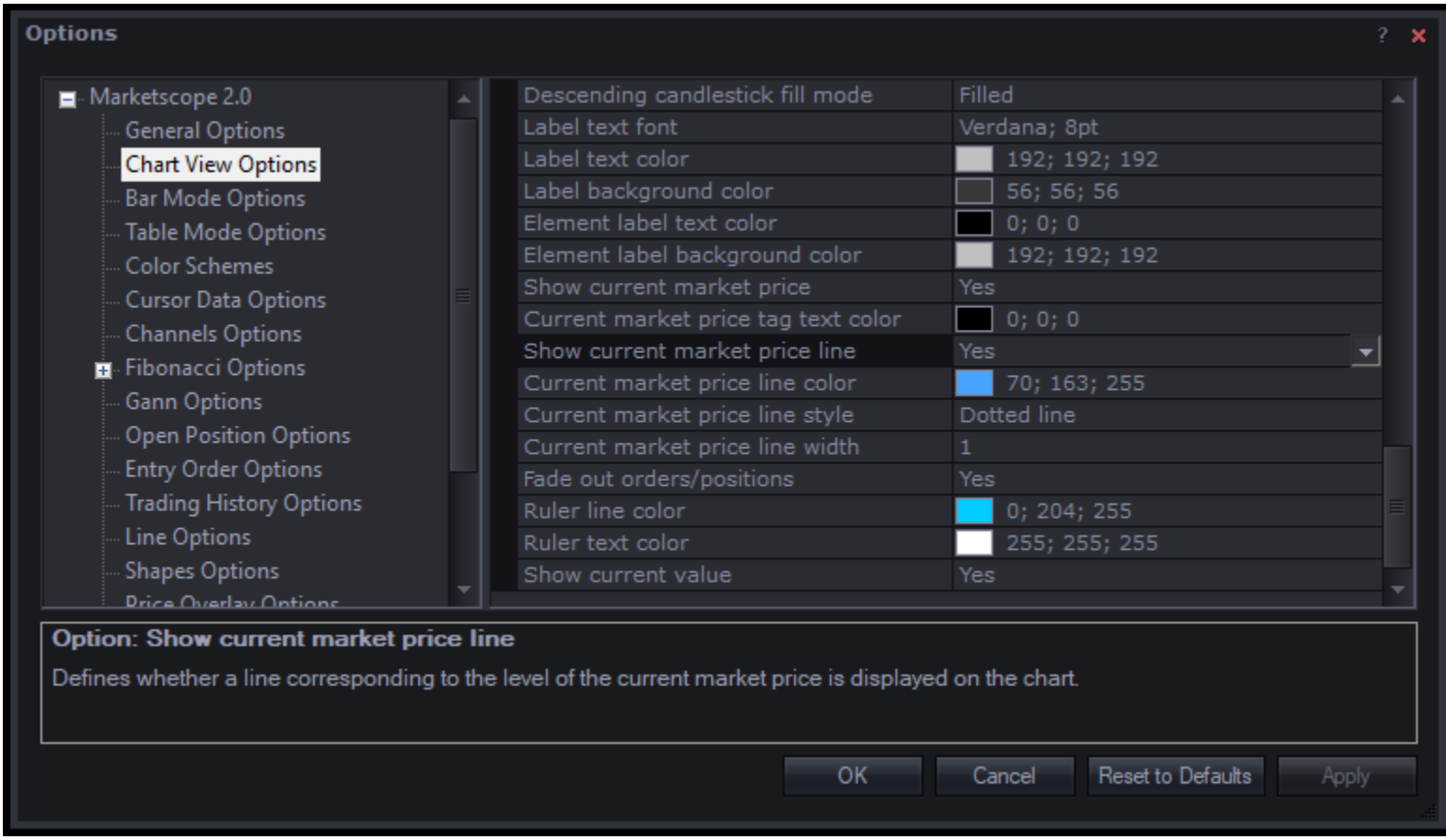
sql-developer

数据库右键：open sql worksheet

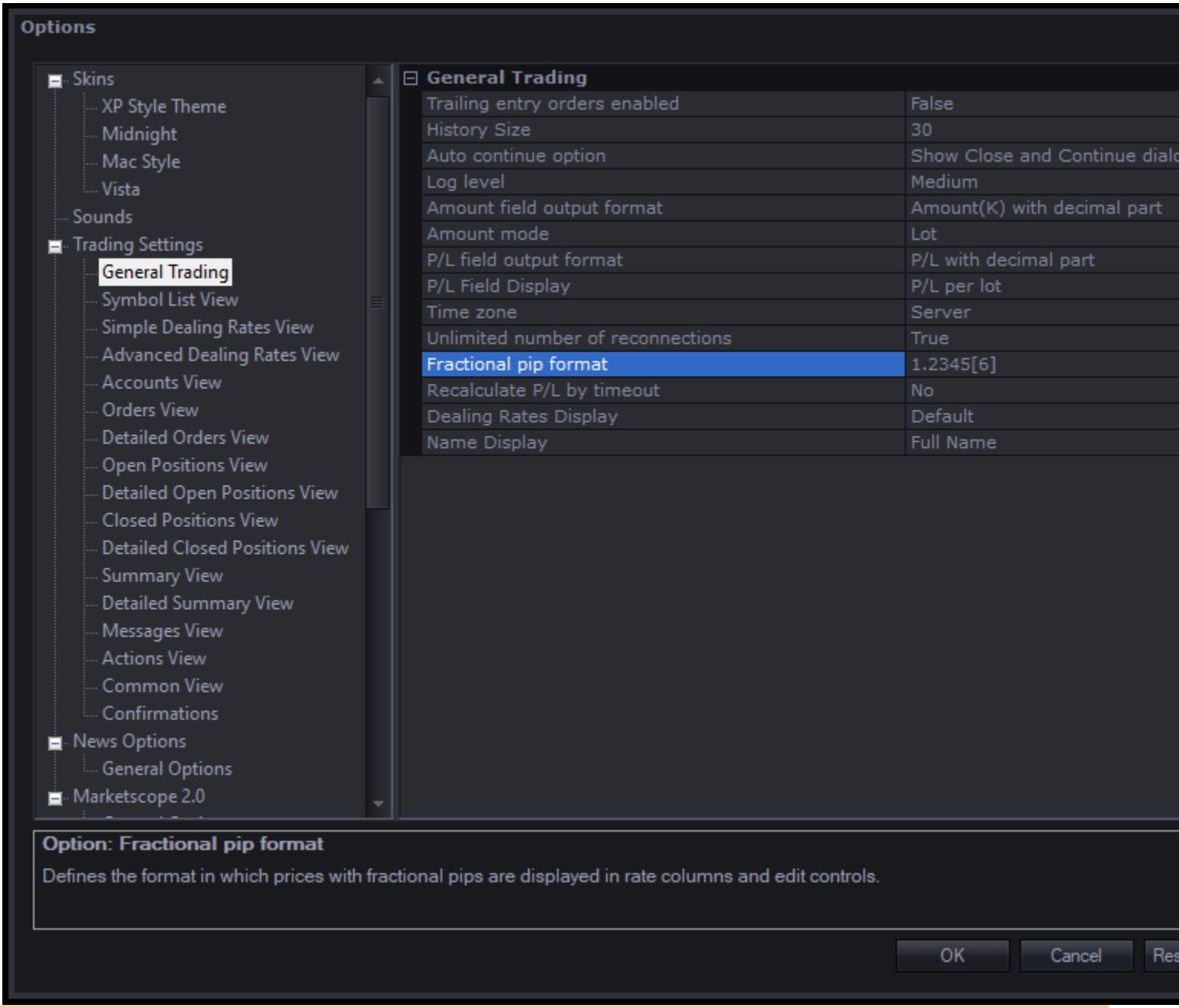
FXCM

Interface Configuration

PriceLine



Fractional Pip Format



Chart

M15　H2　W1

Color

黄色：重点支撑

紫色：交易支撑

tradingView

```
//@version=5


//main page
indicator(title="saidake", shorttitle="SDK", overlay=true, timeframe="", timeframe_gaps=true)

//MA 302
len = input.int(9, minval=1, title="Length")
src = input(close, title="Source")
```

```
offset = input.int(title="Offset", defval=0, minval=-500, maxval=500)
out = ta.sma(src, len)
plot(out, color=color.blue, title="MA", offset=offset)

ma(source, length, type) =>
    switch type
        "SMA" => ta.sma(source, length)
        "EMA" => ta.ema(source, length)
        "SMMA (RMA)" => ta.rma(source, length)
        "WMA" => ta.wma(source, length)
        "VWMA" => ta.vwma(source, length)

typeMA = input.string(title = "Method", defval = "SMA", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"],
group="Smoothing")
smoothingLength = input.int(title = "Length", defval = 5, minval = 1, maxval = 100, group="Smoothing")

smoothingLine = ma(out, smoothingLength, typeMA)
plot(smoothingLine, title="Smoothing Line", color=#f37f20, offset=offset, display=display.none)

//MA 604
len604 = input.int(9, minval=1, title="Length")
src604 = input(close, title="Source")
offset604 = input.int(title="Offset", defval=0, minval=-500, maxval=500)
out604 = ta.sma(src604, len604)
plot(out604, color=color.blue, title="MA", offset=offset604)

ma604(source, length, type) =>
    switch type
        "SMA" => ta.sma(source, length)
        "EMA" => ta.ema(source, length)
        "SMMA (RMA)" => ta.rma(source, length)
        "WMA" => ta.wma(source, length)
        "VWMA" => ta.vwma(source, length)

typeMA604 = input.string(title = "Method", defval = "SMA", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"],
group="Smoothing")
smoothingLength604 = input.int(title = "Length", defval = 5, minval = 1, maxval = 100, group="Smoothing")

smoothingLine604 = ma604(out, smoothingLength604, typeMA604)
plot(smoothingLine604, title="Smoothing Line", color=#f37f20, offset=offset, display=display.none)



//MA 906
len906 = input.int(9, minval=1, title="Length")
src906 = input(close, title="Source")
offset906 = input.int(title="Offset", defval=0, minval=-500, maxval=500)
out906 = ta.sma(src906, len906)
plot(out906, color=color.blue, title="MA", offset=offset906)
```

```
ma906(source, length, type) =>
    switch type
        "SMA" => ta.sma(source, length)
        "EMA" => ta.ema(source, length)
        "SMMA (RMA)" => ta.rma(source, length)
        "WMA" => ta.wma(source, length)
        "VWMA" => ta.vwma(source, length)

typeMA906 = input.string(title = "Method", defval = "SMA", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"],
group="Smoothing")
smoothingLength906 = input.int(title = "Length", defval = 5, minval = 1, maxval = 100, group="Smoothing")

smoothingLine906 = ma906(out, smoothingLength906, typeMA906)
plot(smoothingLine906, title="Smoothing Line", color=#f37f20, offset=offset, display=display.none)
```

typora

Inline LaTeX (for Markdown or LaTeX environments)

{

$$f(i,j) = \begin{cases} cnt[0][j] & i = \\ \max\{cnt[i][v] + f(i-1, j \oplus v)\}, v \in group[i] & i > \end{cases}$$

```
$$
f(i, j) =
\begin{cases}
    cnt[0][j] & i = 0 \\
    \max\{ cnt[i][v] + f(i - 1, j \oplus v) \}, \, v \in group[i] & i > 0
\end{cases}
$$
```

x

Use \cdot for operations resulting in a scalar (dot product).

Use \times for operations resulting in a vector (cross product).t

∀

This symbol represents "for all" and is rendered as the universal quantifier in LaTeX.

$\forall i \in [0, n-k), nums[i] = nums[i+k]$

i ∈ [0, n-k)

This denotes that 'i' is an element of the set of integers from 0 to n-k (inclusive of 0 but exclusive of n-k).

The symbol ∈ in LaTeX represents "element of" or "belongs to".

⊕

$ O(n \log n) $ $$ O(n \log n) $$

$A \oplus B$ ⊕

\frac

2^n

2 to the power of n
$$ (n-1) \times \frac{n}{2} $$
\times is used for the multiplication symbol (×).
\frac{n}{2} is the LaTeX way to write a fraction n2\frac{n}{2}2n.
$\frac{(n-1) \times n}{2}$
$$ (a + b)^n = \sum_{k=0}^n C(n, k) \cdot a^{n-k} \cdot b^k $$
a \times b
symbols
x_1
≈
x \approx 10

\sqrt
The equation has two solutions: x = -b ± √(b² - 4ac).
The square root of x is \(\sqrt{x} \).

Anchor

Manual Reference
My Level 4 Heading {#heading-1}

Some content here...

My Level 4 Heading {#heading-2}

Another content section...

You can now reference the first heading: [Reference to My Level 4 Heading 1](#heading-1)

And the second one: [Reference to My Level 4 Heading 2](#heading-2)

Table of Contents

In Markdown, anchor links for headings are generated by converting the heading text to lowercase, replacing spaces with hyphens, and removing any special characters.
In your case, the anchor for "Directory Structure" should be written as #directory-structure.

- [Introduction](#introduction)
- [Installation](#installation)
  - [System Requirements](#system-requirements)
  - [Installation Steps](#installation-steps)
- [Usage](#usage)
  - [Basic Usage](#basic-usage)
  - [Advanced Usage](#advanced-usage)
- [Configuration](#configuration)
  - [General Settings](#general-settings)
  - [Customization](#customization)
- [Conclusion](#conclusion)

Content

Double Spaces for a Line Break

`code` Single backticks (``) around text are used for inline code

加粗文本 或者 __加粗文本__

斜体文本 或者_斜体文本_

~~删除文本~~

标题

一级标题（注意前方空格）
二级标题
三级标题
四级标题
五级标题
六级标题

一级标题
======

二级标题

列表

无序列表

- Red
- Green
- Blue

* Red
* Green
* Blue

+ Red
+ Green
+ Blue

有序列表

1. Red
2. Green
3. Blue

Reference

> 这是一段引用 //在`>`后面有 1 个空格
>

>　　这是引用的代码块形式　　//在`>`后面有 5 个空格
>
> 代码例子：
>

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

> 一级引用
> > 二级引用
> > > 三级引用

> #### 这是一个四级标题
>
> 1. 这是第一行列表项
> 2. 这是第二行列表项

Link

[Search for Title](document.pdf#search=YourTitleHere)
　　Open with Adobe Reader & Search for a Title
[Go to Title](document.pdf#nameddest=SectionTitle)
　　Link to a Named Destination (if the PDF supports it)
[Go to Section](document.pdf#page=10)
　　Link to a Page in the PDF

图片：　　![图片文本(可忽略)](图片地址)

链接：[]()　　[链接文本](链接地址)

这是行内式链接：[ConnorLin's Blog](http://connorlin.github.io)
这是参考式链接：[ConnorLin's Blog][url]，其中 url 为链接标记，可置于文中任意位置。
[url]: http://connorlin.github.io/ "ConnorLin's Blog"
链接标记格式为：[链接标记文本]:　链接地址　链接 title(可忽略)
这是自动链接：直接使用`<>`括起来<http://connorlin.github.io>
这是图片：![][avatar]
[avatar]: https://connorlin.github.io/images/avatar.jpg

分隔线

***

___

* * *

Table

Column 1	Column 2	Column 3
Row 1	Data 1	Data 2
Row 2	Data 3	Data 4

```
| Row 3    | Data 5   | Data 6   |

| Left-aligned | Center-aligned | Right-aligned |
|:-------------|:--------------:|--------------:|
| Data 1       | Data 2         | Data 3        |
| More data    | More data      | More data     |
```

:--- for left alignment.
:---: for center alignment.
---: for right alignment.

Code

行内代码使用 `代码` 标识，可嵌入文字中
代码块使用 4 个空格或```标识
```
这里是代码
```
代码语法高亮在 ```后面加上空格和语言名称即可
``` 语言
//注意语言前面有空格
这里是代码
```

Html

字体及字体颜色和大小
字体颜色
使用 html 标签`<br/>`
换行
居左文本
居中文本
居右文本
<u>下划线文本</u>

锚点
第一步：添加链接　　[测试 2](#test2)
第二部：添加锚点　　测试 2