部署实施安装指导书

1.部署过程

1. 1. 1. 部署应用

application-dev.yml 文件是所有应用公用的配置信息，根据实际情况修改，配置信息如下：

vi application-dev.yml

```
spring:
    #使用 okhttp 替换 openfeign 的默认实现
  cloud:
    openfeign:
      okhttp:
        enabled: true
    # 关闭负载重试
    loadbalancer:
      retry:
        enabled: false
      rebbion:
        enabled: false
  rabbitmq:
    host: rabbitmq.default
    port: 5672
    username: admin
    password: password
    virtual-host: /
    service:
      host: rabbitmq.default
      port: 5672
      username: service_user
      password: 123456
      virtual-host: service

  data:
    redis:
      host: redis-svc
      port: 6379
      password: password
      database: 1
```

```
        # 读超时
        timeout: 3000
        # 连接超时
        connectTimeout: 5000
        # Lettuce 连接池
        lettuce:
          pool:
            # 最小空闲连接
            min-idle: 5
            # 最大空闲连接
            max-idle: 10
            # 最大连接数
            max-active: 100
            # 连接分配应该阻塞的最大时间
            max-wait: 2000

  mvc:
    contentnegotiation:
      favor-parameter: true
      default-content-type: application/json
      media-types:
        json: application/json
    converters:
      preferred-json-mapper: jackson

# mybatis-plus 配置
mybatis-plus:
  type-handlers-package: com.huizhi.agent.common.typeHandler
  tenant-enable: ture
  mapper-locations: classpath:/mapper/*Mapper.xml,classpath:/mapper/*/*Mapper.xml
  global-config:
    capitalMode: true
    banner: false
    db-config:
      id-type: auto
      select-strategy: not_empty
      insert-strategy: not_empty
      update-strategy: not_null
  configuration:
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl

#feign 的日志配置
logging:
  level:
```

```
        com.huizhi.agent.account.api.feign: DEBUG
        com.huizhi.agent.im.api.loadbalancer: DEBUG
        org.springframework.cloud.loadbalancer: DEBUG


okhttp:
  pool:
  #设置读超时
      connectTimeout: 30
      #设置读超时
      readTimeout: 300
      #设置写超时
      writeTimeout: 300
      # 是否自动重连
      retryOnConnectionFailure: false
      #配置连接池中的最大空闲线程个数
      maxIdleConnections: 5
      #空闲线程保留时长
      keepAliveTime: 30

dict:
  package-names:
      - com.huizhi.agent.prompt.api.enums
      - com.huizhi.agent.model.api.enums
      - com.huizhi.agent.module.api.enums
      - com.huizhi.agent.knowledge.api.enums
      - com.huizhi.agent.agents_app.api.enums
      - com.huizhi.agent.message.api.enums


webclient:
  enabled: true
  #最大连接数
  max-connections: 500
  #最大空闲时间（秒）
  max-idle-time: 20
  #连接超时时间（毫秒）
  connect-timeout: 10000
  #响应时间（秒）
  response-timeout: 10
  #read timeout in seconds
  read-timeout: 10
  #write timeout in seconds
  write-timeout: 10
```

```
mq:
  type: rabbitmq
  rabbitmq:
    topic-queue:
      name: topic.Queue
      exchange: topic.exchange
      routing-key: topic
      max-retries: 3
      max-retries-key: topic:try
    fanout-queue:
      name: fanoutQueue
      exchange: fanout.exchange
      routing-key: fanout
      max-retries: 3
      max-retries-key: fanout:try
    direct-queue:
      name: directQueue
      exchange: direct.exchange
      routing-key: direct
      max-retries: 3
      max-retries-key: direct:try
    ttl-queue:
      name: ttl.queue
      exchange: ttl.direct
      routing-key: ttl
      max-retries: 3
      max-retries-key: ttl:try
    dlx-queue:
      name: dlx.queue
      exchange: dlx.direct
      routing-key: dlx
      max-retries: 3
      max-retries-key: dlx:try

# 登陆相关设置
login:
  # 1、手机号验证码/用户名密码 2、手机号验证码 3、用户名密码
  mode: 1

# 网关解密登录前端密码
gateway:
  encode-key: 'huizhihuyu201707'
```

1.1.1.1. agent-count

1）修改 agent-account-service-sharding-dev.yml 配置文件,配置正确的数据库信息
vi agent-account-service-sharding-dev.yml

```
# 数据源配置
dataSources:
  # 模型库
  ds_account: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    driverClassName: org.postgresql.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_account?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_account?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: root
    #password: password
    username: admin
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4
  # 通用库
  ds_common: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    username: admin
    password: password
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: root
```

```
    #password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4
  # 用户账号相关库
  # ds_account: # 逻辑名称
  #   dataSourceClassName: com.zaxxer.hikari.HikariDataSource
  #   driverClassName: dm.jdbc.driver.DmDriver
  #                                                                                url:
jdbc:dm://127.0.0.1:5236?schema=AGENTS_ACCOUNT&zeroDateTimeBehavior=convertToNu
ll&useUnicode=true&characterEncoding=utf-8
  #   username: SYSDBA
  #   password: Cht005288
  #   hikari:
  #     minimum-idle: 5
  #     maximum-pool-size: 20
  #     connection-test-query: SELECT 1
  #     max-lifetime: 1800000
  #     connection-timeout: 30000
  #     pool-name: DatebookHikariCP
  #     connection-init-sql: set names utf8mb4

  # # 公共库
  # ds_common: # 逻辑名称
  #   dataSourceClassName: com.zaxxer.hikari.HikariDataSource
  #   driverClassName: dm.jdbc.driver.DmDriver
  #                                                                                url:
jdbc:dm://127.0.0.1:5236?schema=AGENTS_COMMON&zeroDateTimeBehavior=convertToNul
l&useUnicode=true&characterEncoding=utf-8
  #   username: SYSDBA
  #   password: Cht005288
  #   hikari:
  #     minimum-idle: 5
  #     maximum-pool-size: 20
  #     connection-test-query: SELECT 1
  #     max-lifetime: 1800000
  #     connection-timeout: 30000
  #     pool-name: DatebookHikariCP
  #     connection-init-sql: set names utf8mb4
```

```
## 规则配置
rules:
  - !ENCRYPT
    encryptors:
      aes_encryptor:
        type: AES
        props:
          aes-key-value: hzhy20170701
      assisted_encryptor:
        type: MD5
        props:
          salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
        type: KEEP_FIRST_N_LAST_M #脱敏算法类型
        props:
          first-n: 3
          last-m: 4
          replace-char: '*'

  - !SINGLE
    tables:          #单表列表设置
      - ds_account.*
      - ds_common.*


props:
  # 是否打印 SQL
  sql-show: true
```

2）编辑 agent-account-service-dev.yml 文件，根据实际情况填写

vi agent-account-service-dev.yml

```
# spring:
#     datasource:
#         type: com.zaxxer.hikari.HikariDataSource
#         driver-class-name: com.mysql.cj.jdbc.Driver
#                                                                                 url:
jdbc:mysql://116.62.172.79:3306/agents_account?useUnicode=true&characterEncoding=utf-8&us
eSSL=false&serverTimezone=Asia/Shanghai
#         username: username
#         password: password
#         hikari:
#             minimum-idle: 5
#             maximum-pool-size: 20
#             connection-test-query: SELECT 1
#             max-lifetime: 1800000
#             connection-timeout: 30000
#             pool-name: DatebookHikariCP
#             connection-init-sql: set names utf8mb4

mobile-message:
  whiteIpList:
    - 58.212.80.109
  realSendWhiteIpList:
    - dev
    - sit
    - pre
  # 手机号注册是否通知用户名/初始密码
  notifyInitPassword: false
```

3）打开编排文件 deploy-agent-account.yaml，替换正确的镜像,其他配置根据实际配置

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-account
  namespace: test-k8s
spec:
```

```
  replicas: 1
  selector:
    matchLabels:
      app: agent-account
  template:
    metadata:
      labels:
        app: agent-account
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agent-account
        image: huizhizhinengoso/agent-account:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "7dd5b55b-0c7e-42fb-9bb0-158a02034494"
```

4）部署服务

```
kubectl apply -f deploy-agent-account.yaml
```

1.1.1.2. agent-agentsapp

1) 修改配置文件 agent-agents-application-service-dev.yml，根据实际情况修写

```
vi agent-agents-application-service-dev.yml
```

```
#雪花算法配置
snowflake:
  enabled: true
  workerId: 1
  datacenterId: 1
apikeysecret: aB3eF7hJ9kLmNoPqRsTuVwXyZ1bC2dE4

apiInterface:
  codeExampleVOS: '[{"bizType":1,"snippets":[{"codeType":1,"normalCode":"api 接口"},{"codeType":2,"normalCode":"api 接口"},{"codeType":3,"normalCode":"api 接口"}]},
  {"bizType":2,"snippets":[{"codeType":1,"normalCode":"api 接口"},{"codeType":2,"normalCode":"api 接口"},{"codeType":3,"normalCode":"api 接口"}]},
```

{"bizType":3,"snippets":[{"codeType":1,"normalCode":"api 接口"},{"codeType":2,"normalCode":"api 接口"},{"codeType":3,"normalCode":"api 接口"}]},
{"bizType":4,"snippets":[{"codeType":1,"normalCode":"api 接口"},{"codeType":2,"normalCode":"api 接口"},{"codeType":3,"normalCode":"api 接口"}]},
{"bizType":5,"snippets":[{"codeType":1,"normalCode":"api 接口"},{"codeType":2,"normalCode":"api 接口"},{"codeType":3,"normalCode":"api 接口"}]},
{"bizType":6,"snippets":[{"codeType":1,"normalCode":"api 接口","fileCode":"文件上传接口"},{"codeType":2,"normalCode":"api 接口","fileCode":"文件上传接口"},{"codeType":3,"normalCode":"api 接口","fileCode":"文件上传接口"}]}]'

2) 修改配置文件 agent-agents-application-service-sharding-dev.yml，根据实际情况填写

vi agent-agents-application-service-sharding-dev.yml

```
# 数据源配置
dataSources:
  ds_agents_application: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_application?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: username
    #password: password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_application?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    username: admin
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

  # 公共库
  ds_common: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
```

```
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
      #username: root
      #password: password
      driverClassName: org.postgresql.Driver
      url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
      username: admin
      password: password
      hikari:
        minimum-idle: 5
        maximum-pool-size: 20
        connection-test-query: SELECT 1
        max-lifetime: 1800000
        connection-timeout: 30000
        pool-name: DatebookHikariCP
        connection-init-sql: set names utf8mb4

## 规则配置
rules:
  - !ENCRYPT
    encryptors:
      aes_encryptor:
        type: AES
        props:
          aes-key-value: hzhy20170701
      assisted_encryptor:
        type: MD5
        props:
          salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
```

```
            type: KEEP_FIRST_N_LAST_M #脱敏算法类型
            props:
              first-n: 3
              last-m: 4
              replace-char: '*'

  - !SINGLE
    tables:          #单表列表设置
      - ds_agents_application.*
      - ds_common.*


props:
  # 是否打印 SQL
  sql-show: true
```

3) 修改编排文件 deploy-agent-agentsapp.yaml，根据实际情况修改

cat deploy-agent-agentsapp.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-agentapp
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-agentapp
  template:
    metadata:
      labels:
        app: agent-agentapp
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agent-agentapp
        image: huizhizhinengoso/agent-agentsapp:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
```

```
            - name: NACOS_PORT
              value: "8848"
            - name: NACOS_NAMESPACE
              value: "7dd5b55b-0c7e-42fb-9bb0-158a02034494"
            - name: ARGS
            ports:
            - containerPort: 6000
```

kubectl apply -f deploy-agent-agentsapp.yaml

1.1.1.3. agent-auth

1) 修改编排文件 deploy-agent-auth.yaml，根据实际情况修改

```
root@k8s-master:/opt/server# cat deploy-agent-auth.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-auth
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-auth
  template:
    metadata:
      labels:
        app: agent-auth
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agent-auth

        image: huizhizhinengoso/agent-auth:v2.1.0.20250612-Release

        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "7dd5b55b-0c7e-42fb-9bb0-158a02034494"
```

```
        ports:
        - containerPort: 5000

##部署服务
kubectl apply -f deploy-agent-auth.yaml
```

1.1.1.4. agent-im

1) 修改 agent-im-service-sharding-dev.yml 配置文件

```
# 数据源配置
dataSources:
  ds_im: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_knowledge?useUnicode=true&character
Encoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: root
    #password: password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_knowledge?useUnicode=true&characterEncoding=u
tf-8&useSSL=false&serverTimezone=Asia/Shanghai
    username: admin
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

  # 公共库
  ds_common: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEn
coding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: root
    #password: password
```

```
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    username: admin
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

## 规则配置
rules:
  - !ENCRYPT
    encryptors:
      aes_encryptor:
        type: AES
        props:
          aes-key-value: hzhy20170701
      assisted_encryptor:
        type: MD5
        props:
          salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
        type: KEEP_FIRST_N_LAST_M #脱敏算法类型
        props:
          first-n: 3
          last-m: 4
```

```
            replace-char: '*'

  - !SINGLE
      tables:          #单表列表设置
        - ds_im.*
        - ds_common.*


props:
  # 是否打印 SQL
  sql-show: true
```

2) 修改编排文件 deploy-agent-im.yaml 镜像为实际版本，其他根据实际调整

```
 apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-im-01
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-im
  template:
    metadata:
      labels:
        app: agent-im
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node2
      containers:
      - name: agent-im1
        image: huizhizhinengoso/agent-agentsim:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "7dd5b55b-0c7e-42fb-9bb0-158a02034494"
        - name: ws.netty.ip
          valueFrom:
            fieldRef:
```

```
                fieldPath: status.hostIP
          - name: ws.netty.port
            value: "8009"
          ports:
          - containerPort: 3000
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-im-02
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-im
  template:
    metadata:
      labels:
        app: agent-im
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node2
      containers:
      - name: agent-im1
        image: huizhizhinengoso/agent-agentsim:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "7dd5b55b-0c7e-42fb-9bb0-158a02034494"
        - name: ws.netty.ip
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
        - name: ws.netty.port
          value: "8010"
        ports:
        - containerPort: 3000
kubectl apply -f deploy-agent-im.yaml
```

1.1.1.5. agent-knowledge

1) 修改配置文件 agent-knowledge-service-dev.yml，根据实际填写

```
# spring:
#     datasource:
#         type: com.zaxxer.hikari.HikariDataSource
#         driver-class-name: com.mysql.cj.jdbc.Driver
#                                                                                                 url:
jdbc:mysql://116.62.172.79:3306/agents_knowledge?useUnicode=true&characterEncoding=utf-8
&useSSL=false&serverTimezone=Asia/Shanghai
#         username: username
#         password:password
#         hikari:
#             minimum-idle: 5
#             maximum-pool-size: 200
#             connection-test-query: SELECT 1
#             max-lifetime: 1800000
#             connection-timeout: 30000
#             pool-name: DatebookHikariCP
#             connection-init-sql: set names utf8mb4
#雪花算法配置
snowflake:
  enabled: true
  workerId: 1
  datacenterId: 1
knowledge-api:
  connect-timeout: 30000
  read-timeout: 5000
  file-callback: https://agentic.dev.agentsyun.com/api/knowledge/third/knowledge/info/callback
  chunk-default-size: 500
```

2) 修改配置文件 agent-knowledge-service-sharding-dev.yml，根据实际填写

```
# 数据源配置
dataSources:
  ds_knowledge: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    driverClassName: org.postgresql.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_knowledge?useUnicode=true&character
Encoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_knowledge?useUnicode=true&characterEncoding=u
tf-8&useSSL=false&serverTimezone=Asia/Shanghai
    username: password
```

```
      password: password
      #username: username
      #password: password
      hikari:
        minimum-idle: 5
        maximum-pool-size: 20
        connection-test-query: SELECT 1
        max-lifetime: 1800000
        connection-timeout: 30000
        pool-name: DatebookHikariCP
        connection-init-sql: set names utf8mb4

  # 公共库
  ds_common: # 逻辑名称
      dataSourceClassName: com.zaxxer.hikari.HikariDataSource
      driverClassName: org.postgresql.Driver
      #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEn
coding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
      url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimezone=Asia/Shanghai
      username: admin
      password: password
      #username: root
      #password: password
      hikari:
        minimum-idle: 5
        maximum-pool-size: 20
        connection-test-query: SELECT 1
        max-lifetime: 1800000
        connection-timeout: 30000
        pool-name: DatebookHikariCP
        connection-init-sql: set names utf8mb4

## 规则配置
rules:
  - !ENCRYPT
      encryptors:
        aes_encryptor:
          type: AES
          props:
            aes-key-value: hzhy20170701
        assisted_encryptor:
```

```
            type: MD5
            props:
              salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
        type: KEEP_FIRST_N_LAST_M #脱敏算法类型
        props:
          first-n: 3
          last-m: 4
          replace-char: '*'

  - !SINGLE
    tables:          #单表列表设置
      - ds_knowledge.*
      - ds_common.*


props:
  # 是否打印 SQL
  sql-show: true
```

3) 修改编排文件 deploy-agent-knowledge.yaml，配置指定版本镜像，根据实际情况配置

```
root@k8s-master:/opt/server# cat deploy-agent-knowledge.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-knowledge
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-knowledge
```

```
  template:
    metadata:
      labels:
        app: agent-knowledge
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agent-knowledge
        image: huizhizhinengoso/agent-knowledge:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "9194ecc4-83f3-4745-84d1-25c9aa1e70d0"
        - name: ARGS
        ports:
        - containerPort: 6000

##执行如下命令部署
kubectl apply -f deploy-agent-knowledge.yaml
```

1.1.1.6. agent-message

1) 修改配置文件 agent-message-service-dev.yml，根据实际情况填写

```
 # #message 服务
# spring:
#     datasource:
#       type: com.zaxxer.hikari.HikariDataSource
#       driver-class-name: com.mysql.cj.jdbc.Driver
#                                                                                 url:
jdbc:mysql://116.62.172.79:3306/agents_message?useUnicode=true&characterEncoding=utf-8&u
seSSL=false&serverTimezone=Asia/Shanghai
#       username: root
#       password: hzhy20170701.
#       hikari:
#         minimum-idle: 5
#         maximum-pool-size: 20
#         connection-test-query: SELECT 1
#         max-lifetime: 1800000
#         connection-timeout: 30000
```

```
#          pool-name: DatebookHikariCP
#          connection-init-sql: set names utf8mb4

#雪花算法配置
snowflake:
  enabled: true
  workerId: 1
  datacenterId: 1
```

2) 修改配置文件 agent-message-service-sharding-dev.yml，根据实际情况填写

```
# 数据源配置
dataSources:
  # 消息库
  ds_message: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_message?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: root
    #password: password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_message?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    username: admin
    password: password
    hikari:
      # 启动时立即初始化连接
      initialization-fail-timeout: 0
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4
  # 通用库
  ds_common: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
```

```
    #username: root
    #password: password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    username: admin
    password: password
    hikari:
      # 启动时立即初始化连接
      initialization-fail-timeout: 0
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

## 规则配置
rules:
  - !ENCRYPT
    encryptors:
      aes_encryptor:
        type: AES
        props:
          aes-key-value: hzhy20170701
      assisted_encryptor:
        type: MD5
        props:
          salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
```

```
            type: KEEP_FIRST_N_LAST_M #脱敏算法类型
            props:
              first-n: 3
              last-m: 4
              replace-char: '*'

  - !SINGLE
    tables:          #单表列表设置
      - ds_message.*
      - ds_common.*


props:
  # 是否打印 SQL
  sql-show: true
```

3) 修改编排文件 deploy-agent-message.yaml，根据实际替换为指定镜像版本

```
    root@k8s-master:/opt/server# cat deploy-agent-message.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-message
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-message
  template:
    metadata:
      labels:
        app: agent-message
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node2
      containers:
      - name: agent-message
        image: huizhizhinengoso/agent-message:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
```

```
            - name: NACOS_NAMESPACE
              value: "9194ecc4-83f3-4745-84d1-25c9aa1e70d0"
            ports:
            - containerPort: 1200
##执行下面部署命令
kubectl apply -f deploy-agent-message.yaml
```

1.1.1.7. agent-model

1) 修改配置文件 agent-model-service-dev.yml，根据实际情况调整

```
    # spring:
#     datasource:
#       type: com.zaxxer.hikari.HikariDataSource
#       driver-class-name: com.mysql.cj.jdbc.Driver
#                                                                                 url:
jdbc:mysql://116.62.172.79:3306/agents_model?useUnicode=true&characterEncoding=utf-8&use
SSL=false&serverTimezone=Asia/Shanghai
#       username: root
#       password: hzhy20170701.
#       hikari:
#         minimum-idle: 5
#         maximum-pool-size: 20
#         connection-test-query: SELECT 1
#         max-lifetime: 1800000
#         connection-timeout: 30000
#         pool-name: DatebookHikariCP
#         connection-init-sql: set names utf8mb4



#雪花算法配置
snowflake:
  enabled: true
  workerId: 1
  datacenterId: 1
model-api-base:
  connect-timeout: 30000
  read-timeout: 5000
```

2) 修改配置文件 agent-model-service-sharding-dev.yml，根据实际情况填写

```
# 数据源配置
dataSources:
  # 模型库
```

```
  ds_model: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_model?useUnicode=true&characterEncoding=utf-8
&useSSL=false&serverTimezone=Asia/Shanghai
    username: admin
    password: password
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_model?useUnicode=true&characterEnco
ding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: root
    #password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4
  # 通用库
  ds_common: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimezone=Asia/Shanghai
    username: admin
    password: password
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEn
coding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: username
    #password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
```

```
        connection-init-sql: set names utf8mb4

## 规则配置
rules:
  - !ENCRYPT
    encryptors:
      aes_encryptor:
        type: AES
        props:
          aes-key-value: hzhy20170701
      assisted_encryptor:
        type: MD5
        props:
          salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
        type: KEEP_FIRST_N_LAST_M #脱敏算法类型
        props:
          first-n: 3
          last-m: 4
          replace-char: '*'

  - !SINGLE
    tables:          #单表列表设置
      - ds_model.*
      - ds_common.*


props:
  # 是否打印 SQL
  sql-show: true
```

3) 修改编排文件 deploy-agent-model.yaml，根据实际替换为指定镜像版本，其他配置根据

实际填写

root@k8s-master:/opt/server# cat deploy-agent-model.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-model
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-model
  template:
    metadata:
      labels:
        app: agent-model
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agent-model
        image: huizhizhinengoso/agent-model:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "9194ecc4-83f3-4745-84d1-25c9aa1e70d0"
        - name: ARGS
        ports:
        - containerPort: 8100

##执行如下命令执行部署
kubectl apply -f deploy-agent-model.yaml
```

1.1.1.8. agent-module

1) 修改配置文件 agent-module-service-dev.yml，根据实际情况填写

```
#雪花算法配置
snowflake:
  enabled: true
  workerId: 1
```

```
  datacenterId: 1
```

2) 修改配置文件 agent-module-service-sharding-dev.yml，根据实际情况填写

```
# 数据源配置
dataSources:
  ds_agents_module: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_module?useUnicode=true&characterEnc
oding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: username
    #password:password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_module?useUnicode=true&characterEncoding=utf-8
&useSSL=false&serverTimezone=Asia/Shanghai
    username: username
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

  # 公共库
  ds_common: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEn
coding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: username
    #password: password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimezone=Asia/Shanghai
    username:username
    password: password
    hikari:
```

```
        minimum-idle: 5
        maximum-pool-size: 20
        connection-test-query: SELECT 1
        max-lifetime: 1800000
        connection-timeout: 30000
        pool-name: DatebookHikariCP
        connection-init-sql: set names utf8mb4

## 规则配置
rules:
  - !ENCRYPT
    encryptors:
      aes_encryptor:
        type: AES
        props:
          aes-key-value: hzhy20170701
      assisted_encryptor:
        type: MD5
        props:
          salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
        type: KEEP_FIRST_N_LAST_M #脱敏算法类型
        props:
          first-n: 3
          last-m: 4
          replace-char: '*'

  - !SINGLE
    tables:          #单表列表设置
      - ds_agents_module.*
      - ds_common.*
```

```
props:
  # 是否打印 SQL
  sql-show: true
```

3) 修改编排文件 deploy-agent-module.yaml，根据实际情况替换为自己镜像

root@k8s-master:/opt/server# cat deploy-agent-module.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-module
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-module
  template:
    metadata:
      labels:
        app: agent-module
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agent-module
        image: huizhizhinengoso/agent-module:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "9194ecc4-83f3-4745-84d1-25c9aa1e70d0"
        - name: ARGS
        ports:
        - containerPort: 6000
```

##执行如下命令执行部署

```
kubectl apply -f deploy-agent-module.yaml
```

1.1.1.9. agent-prompt

1) 修改配置文件 agent-prompt-service-dev.yml，根据实际情况修改

```
# spring:
#   datasource:
#     type: com.zaxxer.hikari.HikariDataSource
#     driver-class-name: com.mysql.cj.jdbc.Driver
#     url: jdbc:mysql://116.62.172.79:3306/agents_prompt?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
#     username: root
#     password: hzhy20170701.
#     hikari:
#       minimum-idle: 5
#       maximum-pool-size: 20
#       connection-test-query: SELECT 1
#       max-lifetime: 1800000
#       connection-timeout: 30000
#       pool-name: DatebookHikariCP
#       connection-init-sql: set names utf8mb4

prompt:
  #查询提示词的长度
  length: 20

#雪花算法配置
snowflake:
  enabled: true
  workerId: 1
  datacenterId: 1
```

2) 修改配置文件 agent-prompt-service-sharding-dev.yml，根据实际情况修改

```
# 数据源配置
dataSources:
  ds_prompt: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url: jdbc:mysql://mysql-source-headless.default:3306/agents_prompt?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: username
    #password: password
    driverClassName: org.postgresql.Driver
```

```
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_prompt?useUnicode=true&characterEncoding=utf-8
&useSSL=false&serverTimezone=Asia/Shanghai
    username: username
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

  # 公共库
  ds_common: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEn
coding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: username
    #password: password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-
8&useSSL=false&serverTimezone=Asia/Shanghai
    username: username
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

## 规则配置
rules:
  - !ENCRYPT
    encryptors:
      aes_encryptor:
        type: AES
```

```
        props:
          aes-key-value: hzhy20170701
      assisted_encryptor:
        type: MD5
        props:
          salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
        type: KEEP_FIRST_N_LAST_M #脱敏算法类型
        props:
          first-n: 3
          last-m: 4
          replace-char: '*'

  - !SINGLE
    tables:          #单表列表设置
      - ds_prompt.*
      - ds_common.*


props:
  # 是否打印 SQL
  sql-show: true
```

3) 修改编排文件 deploy-agent-prompt.yaml，替换为自己的实际镜像

```
root@k8s-master:/opt/server# cat deploy-agent-prompt.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-prompt
  namespace: test-k8s
spec:
  replicas: 1
```

```
  selector:
    matchLabels:
      app: agent-prompt
  template:
    metadata:
      labels:
        app: agent-prompt
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agent-prompt
        image: huizhizhinengoso/agent-prompt:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "9194ecc4-83f3-4745-84d1-25c9aa1e70d0"
        - name: ARGS
        ports:
        - containerPort: 7000
```

##执行如下命令安装部署

```
kubectl apply -f deploy-agent-prompt.yaml
```

1.1.1.10. agent-workflow

1) 修改配置文件 agent-workflow-service-dev.yml，根据实际情况填写

```
    spring:
  http:
    encoding:
      charset: UTF-8
      enabled: true
      force: true

rabbitmq:
  workflow:
    max-retries: 3
    max-retries-key: rabbitmq:workflow:try

#雪花算法配置
```

```
snowflake:
  enabled: true
  workerId: 1
  datacenterId: 1
```

2) 修改配置文件 agent-workflow-service-dev.yml，根据实际情况填写

```
# 数据源配置
dataSources:
  ds_workflow: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_workflow?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username:username
    #password: password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_workflow?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    username: username
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

  # 公共库
  ds_common: # 逻辑名称
    dataSourceClassName: com.zaxxer.hikari.HikariDataSource
    #driverClassName: com.mysql.cj.jdbc.Driver
    #url:
jdbc:mysql://mysql-source-headless.default:3306/agents_common?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai
    #username: username
    #password: password
    driverClassName: org.postgresql.Driver
    url:
jdbc:postgresql://pg-sql.test-k8s:5432/agents_common?useUnicode=true&characterEncoding=utf-
```

```
8&useSSL=false&serverTimezone=Asia/Shanghai
    username: username
    password: password
    hikari:
      minimum-idle: 5
      maximum-pool-size: 20
      connection-test-query: SELECT 1
      max-lifetime: 1800000
      connection-timeout: 30000
      pool-name: DatebookHikariCP
      connection-init-sql: set names utf8mb4

## 规则配置
rules:
  - !ENCRYPT
    encryptors:
      aes_encryptor:
        type: AES
        props:
          aes-key-value: hzhy20170701
      assisted_encryptor:
        type: MD5
        props:
          salt: hzhy20170701

  - !MASK
    maskAlgorithms:
      password_mask: #脱敏算法名称
        type: MD5
        props:
          salt: xyz
      email_mask:
        type: MASK_BEFORE_SPECIAL_CHARS #脱敏算法类型
        props:
          special-chars: '@'
          replace-char: '*'
      phone_mask:
        type: KEEP_FIRST_N_LAST_M #脱敏算法类型
        props:
          first-n: 3
          last-m: 4
          replace-char: '*'

  - !SINGLE
```

```
      tables:          #单表列表设置
        - ds_workflow.*
        - ds_common.*


props:
  # 是否打印 SQL
  sql-show: true
```

3) 修改编排文件 deploy-agent-workflow.yaml，替换为指定版本镜像，其他根据实际情况修改

```
root@k8s-master:/opt/server# cat deploy-agent-workflow.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agent-workflow
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agent-workflow
  template:
    metadata:
      labels:
        app: agent-workflow
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agent-workflow
        image: huizhizhinengoso/agent-workflow:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: NACOS_HOST
          value: "nacos-svc-headless"
        - name: NACOS_PORT
          value: "8848"
        - name: NACOS_NAMESPACE
          value: "9194ecc4-83f3-4745-84d1-25c9aa1e70d0"
        - name: ARGS
        ports:
        - containerPort: 1100
##执行以下命令部署
```

kubectl apply -f deploy-agent-workflow.yaml

1.1.2. python 服务部署

1.1.2.1. mcp-db-access

1) 修改编排文件 db-access-deploy.yaml，根据实际情况替换为镜像

root@k8s-master:/opt/python# cat db-access-deploy.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mcp-db-access
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mcp-db-access
  template:
    metadata:
      labels:
        app: mcp-db-access
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node2
      containers:
      - name: mcp-db-access
        image: huizhizhinengoso/mcp-db-access-server:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: EtcdHosts
          value: "etcd-svc:2379"
        - name: RedisHost
          value: "redis-svc:6379"
        - name: RedisType
          value: "node"
        - name: RedisPass
          value: "password"
        - name: ListenOn
          value: "0.0.0.0:8080"
        - name: DataSource
          value: "postgres://admin:password@pg-sql.test-k8s:5432/mcp?sslmode=disable"
```

##执行以下命令部署
kubectl apply -f db-access-deploy.yaml

1.1.2.2. mcp-gateway

1) 修改编排文件 mcp-gateway.yaml，根据实际替换为自己的镜像

```
root@k8s-master:/opt/python# cat mcp-gateway.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mcp-gateway
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mcp-gateway
  template:
    metadata:
      labels:
        app: mcp-gateway
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node2
      containers:
      - name: mcp-gateway
        image: buffgpt/agentic/mcp-gateway-server:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: EtcdHosts
          value: "etcd-svc:2379"
        - name: RedisHost
          value: "redis-svc:6379"
        - name: RedisType
          value: "node"
        - name: RedisPass
          value: "password"
        - name: Port
          value: "8888"
        - name: Timeout
          value: "10000"
```

##执行如下命令部署
kubectl apply -f mcp-gateway.yaml

1.1.2.3. mcp-validation

1) 修改编排文件 validation-deploy.yaml，根据实际情况替换为指定镜像

```
root@k8s-master:/opt/python# cat validation-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mcp-validation
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mcp-validation
  template:
    metadata:
      labels:
        app: mcp-validation
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node2
      containers:
      - name: mcp-validation
        image: huizhizhinengoso/mcp-validation-server:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: EtcdHosts
          value: "etcd-svc:2379"
        - name: RedisHost
          value: "redis-svc:6379"
        - name: RedisType
          value: "node"
        - name: RedisPass
          value: "password"
        - name: ListenOn
          value: "0.0.0.0:8081"
##执行如下命令部署
kubectl apply -f validation-deploy.yaml
```

1.1.2.4. mcprouter

1) 修改编排文件 mcprouter-deploy.yaml，根据实际情况替换为指定镜像

```
root@k8s-master:/opt/python# cat mcprouter-deploy.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mcprouter
  namespace: test-k8s
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mcprouter
  template:
    metadata:
      labels:
        app: mcprouter
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node2
      containers:
      - name: mcprouter
        image: huizhizhinengoso/mcp-router-server:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        env:
        - name: EtcdHosts
          value: "etcd-svc:2379"
        - name: RedisHost
          value: "redis-svc:6379"
        - name: RedisType
          value: "node"
        - name: RedisPass
          value: "password"
        - name: Port
          value: "8025"
      #    ports:
       # - containerPort: 8025

##执行如下命令部署
kubectl apply -f mcprouter-deploy.yaml
```

1.1.2.5. aiplatform

1) 修改编排文件 aiplatform-daemonset.yaml,服务配置根据实际情况修改

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: aiplatform-config
  namespace: test-k8s
data:
  config.ini: |
    [base_config]
    host = 0.0.0.0
    port = 9991
    workers = 1

    [redis_config]
    password = password
    host = redis-svc
    port = 6379

    [mq_config]
    host = rabbitmq.default
    port = 5672
    user = username
    password = password
    max_priority = 10

    [postgres_config]
    host = pg-sql
    port = 5432
    user = username
    password = password
    database = platform_k8s
    max_connections = 100
    stale_timeout = 300
    timeout = 30

    [my_milvus_config]
    host = 192.168.3.203
    port = 31102
    user = username
    password = password
    alias = default
    protocol = http

    [milvus_config]
    milvus_uri = http://192.168.3.203:31102
    milvus_user = username
    milvus_password = password
```

```
[pipeline_config]
host = 0.0.0.0
port = 8900
workers = 1

[agents_config]
host = 0.0.0.0
port = 8800
workers = 4

[chat_config]
debug = true
position_tool_pins=
position_tool_includes=
position_tool_excludes=

position_provider_pins=
position_provider_includes=
position_provider_excludes=

etl_type=Unstructured
unstructured_api_url=
unstructured_api_key=
scarf_no_analytics=true
prompt_provider=tongyi
prompt_name=qwen-turbo-v2.1.0.20250612-Release
prompt_auth=c4a7b439-26fd-4b51-ad25-ba9877358739

[minio_config]
endpoint = 192.168.3.203:31101
access_key = minioadmin
secret_key = minioadmin

[file_sys_config]
base_url = http://192.168.3.203:31101/

[code_exec_base_config]
base_url = http://127.0.0.1:15685/api/v1/submit

[web_search_config]
default_tool_name = bocha
searxng_url = http://172.16.1.13:8081
```

```
    bocha_url = https://api.bochaai.com/v1/web-search
    bocha_api_key = sk-00514e1469c74b1eb10ce2182ec693ec

    [xinference_config]
    base_url = http://120.26.167.137:9997

    [workflow_config]
    auto_finish_minute = 10

    [mq_name_config]
    knowledge_mq_name = knowledge
    workflow_mq_name = workflowQueue
    workflow_name = workflow
    workflow_application_mq_name = workflowApplicationQueue
    workflow_application_name = workflowApplication
    agents_mq_name = agentQueue
    workflow_chat_queue = workflowChatQueue
    workflow_application_mq_next_node_name = workflowApplicationNextNodeQueue

    [rules_threshold_config]
    threshold = 0.7

    [db_manager_config]
    db_server_url = http://127.0.0.1:8001/api/v1

    [result2excel_config]
    result2excel_server_url = http://127.0.0.1:6070/excel/queryResultToExcel

    [golang_mcp_config]
    golang_mcp_host = http://172.16.11.91:8888

---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: aiplatform-daemonset
  labels:
    app: aiplatform-daemonset
  namespace: test-k8s
spec:
  updateStrategy:
    type: RollingUpdate
  selector:
    matchLabels:
```

```
        app: aiplatform-daemonset
    template:
      metadata:
        labels:
          app: aiplatform-daemonset
      spec:
        dnsPolicy: ClusterFirst
        tolerations:
        - key: nvidia.com/gpu
          operator: Exists
          effect: NoSchedule
        nodeSelector:
          kubernetes.io/hostname: k8s-node3
        containers:
          - name: aiplatform
            image: huizhizhinengoso/aiplatform:v2.1.0.20250612-Release
            imagePullPolicy: IfNotPresent
            ports:
              - name: http
                containerPort: 9991
                hostPort: 9991
                protocol: TCP
            resources:
              limits:
                nvidia.com/gpu.shared: 1
            volumeMounts:
              - name: config
                mountPath: /root/lib/config/config.ini
                subPath: config.ini
        restartPolicy: Always
        volumes:
          - name: config
            configMap:
              name: aiplatform-config
              defaultMode: 0755
##执行如下命令进行部署
kubectl apply -f aiplatform-daemonset.yaml
```

1. 1. 3. 前端部署

1.1.3.1. 前端 web

1）修改前端配置文件 cm.yaml，根据需要调整配置

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-config
  namespace: test-k8s
data:
  nginx.conf: |
    user nginx;
    worker_processes auto;
    error_log /var/log/nginx/error.log;
    pid /run/nginx.pid;
    include /usr/share/nginx/modules/*.conf;

    events {
      worker_connections 1024;
    }

    http {
      log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

      access_log /var/log/nginx/access.log main;

      sendfile                on;
      tcp_nopush              on;
      tcp_nodelay             on;
      keepalive_timeout       65;
      types_hash_max_size 4096;
      include                 /etc/nginx/mime.types;
      default_type            application/octet-stream;

     # include /etc/nginx/conf.d/*.conf;
      # WebSocket 上游服务器配置
      upstream websocket {
        least_conn;   # 使用最少连接数算法
        server agent-im-01.test-k8s.svc.cluster.local:8009 max_fails=99999 fail_timeout=10s;
        server agent-im-02.test-k8s.svc.cluster.local:8010 max_fails=99999 fail_timeout=10s;
        keepalive 32;
        keepalive_requests 100;
        keepalive_timeout 60;
        least_conn;      # 如果 ip_hash 失效，使用最少连接
        }
```

```
upstream file_upload_servers {
    server agent-agentapp.test-k8s:6000 max_fails=3 fail_timeout=30s;
    server agent-prompt.test-k8s:7000 max_fails=3 fail_timeout=30s;
    #server 127.0.0.1:6000 max_fails=3 fail_timeout=30s;
    #server 127.0.0.1:7000 max_fails=3 fail_timeout=30s;
    #server 127.0.0.1:8000 max_fails=3 fail_timeout=30s;
    server minio-svc.test-k8s:9000 max_fails=3 fail_timeout=30s;
  #server 127.0.0.1:1100 max_fails=3 fail_timeout=30s;
  }
upstream api {
    #server    192.168.3.204:30349 max_fails=3 fail_timeout=30s;
    server    gateway.test-k8s.svc.cluster.local:7349 max_fails=3 fail_timeout=30s;
    keepalive 16;
  }
upstream file_upload_servers_apikey {
        server agent-agentapp.test-k8s.svc.cluster.local:6000 max_fails=3 fail_timeout=30s;
        #server 127.0.0.1:6000 max_fails=3 fail_timeout=30s;
    }



server {
    listen 80;
    server_name 192.168.3.204;
    root /usr/share/nginx/html;

    location /ws/ {
# 添加响应头显示后端信息
        add_header X-Debug-Backend $upstream_addr;
        add_header X-Debug-Status $upstream_status;
        proxy_pass http://websocket;  # 移除末尾的 /ws，让路径完整传递到后端
    # 添加错误处理
        proxy_next_upstream error timeout http_500 http_502 http_503 http_504;
        proxy_next_upstream_tries 3;
        proxy_next_upstream_timeout 10s;
    # WebSocket 设置
        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-For $proxy_add_x_forwarded_for;
    # 增加超时时间
        proxy_connect_timeout 10s;
```

```
            proxy_read_timeout 86400s;
            proxy_send_timeout 60s;
        # 添加详细日志
            error_log /var/log/nginx/websocket_error.log debug;}


         # 文件上传配置
        location ~* ^/api/.*/file/upload$ {
                charset utf-8;
                client_body_buffer_size 128k;
                client_max_body_size 100m;   # 上传文件大小限制
                client_body_timeout 300s;    # 上传超时时间
                rewrite ^/api/.*/(file/upload)$ /$1 break;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_connect_timeout 60s;
                proxy_send_timeout 300s;
                proxy_read_timeout 300s;
                proxy_pass http://file_upload_servers;
         }

    #location    .*/uploadFile$ {
        location ~ /.*uploadFile$ {
                charset utf-8;
                client_body_buffer_size 128k;
                client_max_body_size 100m;   # 上传文件大小限制
                client_body_timeout 300s;    # 上传超时时间
                rewrite ^/api/.*/(file/upload)$ /$1 break;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_connect_timeout 60s;
                proxy_send_timeout 300s;
                proxy_read_timeout 300s;
                proxy_pass http://file_upload_servers_apikey;}

    # API 配置
        location /api {
                proxy_redirect off;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
```

```
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_connect_timeout 60s;
                proxy_send_timeout 60s;
                proxy_read_timeout 60s;
                proxy_pass http://api/api;
            # 错误处理
                proxy_intercept_errors on;
                error_page 502 504 /502.html;}

            location / {
               try_files $uri $uri/ /index.html;}
            location ~ \.js$ {
               add_header Content-Type application/javascript;}
             # 静态文件配置
            location /workflow {
               alias /usr/share/nginx/html/;}
         }
      }
##执行如下命令部署
kubectl apply -f cm.yaml
```

2) 修改 k8s 编排文件 web.yaml，根据需要调整配置

```
root@k8s-master:/opt/frontserver/web# cat web.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
   name: agentic-web
   namespace: test-k8s
   labels:
        app: agentic-web
spec:
   replicas: 1
   selector:
      matchLabels:
         app: agentic-web
   template:
      metadata:
         labels:
            app: agentic-web
      spec:
         nodeSelector:
            kubernetes.io/hostname: k8s-node1
         containers:
```

```
        - name: agentic-web
          image: huizhizhinengoso/agentic_web:v2.1.0.20250612-Release
          imagePullPolicy: IfNotPresent
          ports:
          - containerPort: 80
          volumeMounts:
          - name: nginx-config
            mountPath: /etc/nginx/nginx.conf
            subPath: nginx.conf
        volumes:
          - name: nginx-config
            configMap:
              name: nginx-config


---
apiVersion: v1
kind: Service
metadata:
  name: agentic-web
  namespace: test-k8s
  labels:
    app: agentic-web
spec:
  type: NodePort
  selector:
    app: agentic-web    #
  ports:
    - name: agentic-web
      port: 80
      targetPort: 80
      nodePort: 30080
##执行如下命令部署
kubectl apply -f web.yaml

##前端方位地址
http://127.0.0.1:30080
```

1.1.3.2. 后管 boss

1) 修改后管 boss 配置文件 cm.yaml，根据需要调整配置

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: nginx-config-admin
  namespace: test-k8s
data:
  nginx.conf: |
    user nginx;
    worker_processes auto;
    error_log /var/log/nginx/error.log;
    pid /run/nginx.pid;
    include /usr/share/nginx/modules/*.conf;

    events {
      worker_connections 1024;
    }

    http {
      log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

      access_log /var/log/nginx/access.log main;

      sendfile                on;
      tcp_nopush              on;
      tcp_nodelay             on;
      keepalive_timeout       65;
      types_hash_max_size 4096;
      include                 /etc/nginx/mime.types;
      default_type            application/octet-stream;

     # include /etc/nginx/conf.d/*.conf;
    # include /etc/nginx/conf.d/*.conf;
      # WebSocket 上游服务器配置
      upstream websocket {
        least_conn;  # 使用最少连接数算法
        server agent-im-01.test-k8s.svc.cluster.local:8009 max_fails=99999 fail_timeout=10s;
        server agent-im-02.test-k8s.svc.cluster.local:8010 max_fails=99999 fail_timeout=10s;
        keepalive 32;
        keepalive_requests 100;
        keepalive_timeout 60;
        least_conn;      # 如果 ip_hash 失效，使用最少连接
        }
      upstream file_upload_servers {
        server 127.0.0.1:6000 max_fails=3 fail_timeout=30s;
```

```
    server 127.0.0.1:7000 max_fails=3 fail_timeout=30s;
    server 127.0.0.1:8000 max_fails=3 fail_timeout=30s;
    server 127.0.0.1:9000 max_fails=3 fail_timeout=30s;
   #server 127.0.0.1:1100 max_fails=3 fail_timeout=30s;
  }
upstream api {
    #server    192.168.3.204:30349 max_fails=3 fail_timeout=30s;
    server    gateway.test-k8s.svc.cluster.local:7349 max_fails=3 fail_timeout=30s;
    keepalive 16;
  }
upstream file_upload_servers_apikey {
        server 127.0.0.1:6000 max_fails=3 fail_timeout=30s;
    }


server {
    listen 80;
    server_name localhost;
    #root /usr/share/nginx/html;
    #index index.html;

    location ^~/ws {
            proxy_pass http://websocket/ws;
    # 代理到上面的地址去
            proxy_read_timeout 60s;
            proxy_set_header Host $host;
            proxy_set_header X-Real_IP $remote_addr;
            proxy_set_header X-Forwarded-for $remote_addr;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'Upgrade'; }

    location ^~/api {
            proxy_redirect off;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_pass http://api/api; }

    location / {
      alias /usr/share/nginx/html/;
      index index.html;
      try_files $uri $uri/ /index.html;
    }
```

```
        location /static/ {
            alias /usr/share/nginx/html/static/;}

        }
    }
```

##执行如下命令部署

kubectl apply -f cm.yaml

2) 修改 k8s 编排文件 web.yaml，根据需要调整配置

root@k8s-master:/opt/frontserver/admin# cat web.yaml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: agentic-admin
  namespace: test-k8s
  labels:
    app: agentic-admin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: agentic-admin
  template:
    metadata:
      labels:
        app: agentic-admin
    spec:
      nodeSelector:
        kubernetes.io/hostname: k8s-node1
      containers:
      - name: agentic-admin
        image: huizhizhinengoso/agentic_admin_web:v2.1.0.20250612-Release
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
        volumeMounts:
        - name: nginx-config-admin
          mountPath: /etc/nginx/nginx.conf
          subPath: nginx.conf
      volumes:
        - name: nginx-config-admin
          configMap:
```

```
        name: nginx-config-admin


---
apiVersion: v1
kind: Service
metadata:
  name: agentic-admin
  namespace: test-k8s
  labels:
    app: agentic-admin
spec:
  type: NodePort
  selector:
    app: agentic-admin
  ports:
    - name: agentic-admin
      port: 80
      targetPort: 80
      nodePort: 30081

##执行如下命令部署
kubectl apply -f web.yaml

##后管方访问地址
http://127.0.0.1:30081
```

1.1.4. 备注

当主程序启动完执行如下 sql:

```sql
-- ----------------------------
-- Table structure for t_model_support
-- ----------------------------
DROP TABLE IF EXISTS "public"."t_model_support";
CREATE TABLE "public"."t_model_support" (
  "id" "pg_catalog"."uuid" NOT NULL,
  "model_name" "pg_catalog"."varchar" COLLATE "pg_catalog"."default" NOT NULL,
  "provider_id" "pg_catalog"."varchar" COLLATE "pg_catalog"."default",
  "model_specs" "pg_catalog"."varchar" COLLATE "pg_catalog"."default",
  "model_type" "pg_catalog"."varchar" COLLATE "pg_catalog"."default" NOT NULL,
  "max_length" "pg_catalog"."int4",
  "deploy_type" "pg_catalog"."varchar" COLLATE "pg_catalog"."default" NOT NULL,
```

```
  "api_type" "pg_catalog"."int4" NOT NULL,
  "deploy_properties" "pg_catalog"."varchar" COLLATE "pg_catalog"."default",
  "status" "pg_catalog"."int4" NOT NULL,
  "create_time" "pg_catalog"."timestamp" NOT NULL,
  "update_time" "pg_catalog"."timestamp" NOT NULL,
  "fc_status" "pg_catalog"."int4" NOT NULL DEFAULT 0
)
;


-- ----------------------------
-- Records of t_model_support
-- ----------------------------
INSERT INTO "public"."t_model_support" VALUES ('f89197fe-a1b9-4e8c-8841-13b2e4cc8167', 'qw
0.7, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('97c04288-1be2-498c-88f7-62be62d8aa13', 'qw
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
INSERT INTO "public"."t_model_support" VALUES ('bdf1073f-a484-4252-882c-f2a1997f8da0', 'qw
0.7, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('8d86830e-dabd-4e67-a3b3-3e0a49e9834e', 'q
"default": 0.7, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；
性越弱；数值越大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2
INSERT INTO "public"."t_model_support" VALUES ('16445798-dd8c-4ec1-8ec0-a266abd2dfc6', 'qwen
0, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('da5c96cb-1cb3-4652-86cd-9052dea4b95b', 'qwe
0, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('246bbfc7-4132-4560-b660-9066083b2f88', 'qwe
0, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('5b58b9a2-80dc-4fee-a757-718170a9ff9c', 'qwe
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
INSERT INTO "public"."t_model_support" VALUES ('a19a02ed-4115-49e1-a35f-3a938e7eb57e', 'qw
0.7, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('8cac8c6d-53a9-4e0e-8b67-2d578dff5023', 'qwe
0.7, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
```

```
INSERT INTO "public"."t_model_support" VALUES ('29d39527-e2e8-4c6a-8129-33c277aed6eb', 'qw
"max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('20c3cd01-19d9-4724-a3c4-f9e787276e44', 'qw
"max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('5cab9990-8a8b-40f8-ad00-531b8dce1239', 'qw
2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性",
和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-06
INSERT INTO "public"."t_model_support" VALUES ('9c54e967-9419-49c8-8aaa-c8021fcd166d', 'qw
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('609f7a97-8734-4e8f-9d54-63f1c7d10368', 'qw
0.7, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('e30c6e7d-dfc7-44f0-b80c-5536d73474d0', 'qw
2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性",
和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-06
INSERT INTO "public"."t_model_support" VALUES ('c0b63928-27d0-4817-989a-e8b938867a9c', 'qw
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('ea7b7e83-1fab-4315-a82f-f01d05bbb0e1', 'qw
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('db88c871-856e-4c41-9f73-fa24e1c8e530', 'qwe
2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性",
和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-06
INSERT INTO "public"."t_model_support" VALUES ('5dc3f258-f224-4cc3-855a-12e6f9d2320d', 'qw
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('318520a6-47bc-43d2-8a93-f67cca468326', 'en
"min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('24383cd0-0a9f-4b1f-928f-db219f79765e', 'qwe
0, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('7d6af394-c41d-4e17-9361-d1f9ad645fcb', 'qw
"min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('25462ae0-8ee6-47f7-b1da-9e4d2a2aaf75', 'Dou
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('ac1929c1-87e0-4020-96f6-e36d5ae58669', 'qwe
```

```
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66
INSERT INTO "public"."t_model_support" VALUES ('a1d54fcf-1b1b-4444-ba65-fd62e5631b96', 'qw
0.7, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('5aeb71c8-0d82-4f7a-8599-87ebf2fca622', 'gl
"help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性", "key
temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('358caf27-6dd5-470d-adbc-4a71ca04f1ce', 'Mo
1, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('c6e06b7e-b4a1-448a-a19a-f62c0ef0edf0', 'GL
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('acf9e974-03cc-4b98-9e04-8f665343d288', 'er
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('ca03636a-d5d6-42d1-b2b5-db1105966a6a', 'e
"min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('fbc3304c-da29-4c44-8364-582a940daac3', 'er
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
INSERT INTO "public"."t_model_support" VALUES ('eac54fc2-1f0a-4d69-b4d4-c90c72818813', 'ern
0.95, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越
越大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04
INSERT INTO "public"."t_model_support" VALUES ('35201331-2aff-4218-b0ce-d3cd2e09cf07', 'gp
"min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('3d12253f-e7bd-40d0-9728-774cd627198b', 'gp
"min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('2522f6cc-e037-433e-8a17-21a99bae2d07', 'gp
"max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('1163a8d6-d6a1-4831-9bc0-8fb1a49f0d65', 'gpt
1.0, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('e29ae00b-b832-4765-8ead-52aa52d71b9f', 'e
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('96cfca2f-53c3-47c6-b48a-dc1d79c9f39a', 'e
"min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
```

```
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('b9954da5-ff16-4a79-bdea-56e590b156a2', 'e
0.8, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('eecf3443-363c-4d06-86e9-c4a13e856835', 'er
1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性",
和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-04
INSERT INTO "public"."t_model_support" VALUES ('938098a7-bbab-4808-a8a0-12837e55953d', 'er
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
INSERT INTO "public"."t_model_support" VALUES ('96987a5c-9cd5-431e-a06b-d86cda748b87', 'er
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('7ffa6874-43f3-4b9d-ba4d-56fb7ebc285c', 'err
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
INSERT INTO "public"."t_model_support" VALUES ('8ca94364-4bf1-430d-a943-db8e98cc853c', 'g
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('a9d6d48b-c4fe-4e53-b799-fb45b555dafe', 'y
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('ea90183f-991e-46ab-bc18-f338d4e5ecb7', 'er
"min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('d4effbbc-5804-4932-b1ad-a5941fdfd94c', 'gl
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('0bc50221-60a6-4f03-a94a-911fca74c059', 'gl
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('70b3caae-08fc-48e2-a5dc-73afb7f93a86', 'gl
"help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性", "key
temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('5cd10503-84aa-486f-8069-38e135d935a8', 'g
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('2c475b1f-3d0f-4864-8ed4-60374a2042e4', 'cl
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
INSERT INTO "public"."t_model_support" VALUES ('95c920e6-3220-4f2d-b0f7-e2c158e6a541', 'cl
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
```

```
INSERT INTO "public"."t_model_support" VALUES ('e2d4d1a5-6ad8-400e-9354-45cae3327f8c', 'cha
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('77819126-b5e3-4266-9234-320b08b7b913', 'ch
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('cb83d3be-fded-4a4d-a0cd-e95a2414c7bc', 'h
"min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('58c968dd-3ec2-452e-87ad-db2d587d0bff', 'hun
1.0, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('e963639e-d928-4947-94af-f20869ae7e7b', 'hu
"max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('03e33d20-2fba-4e5c-9d7e-04bf5e511a02', 'hu
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('5793d5e9-4c87-41c9-99c9-16802ea685ed', 'mo
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('d9353eb4-2018-4871-bcf4-26b694f09411', 'qw
"min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('a1665efc-c64c-482d-8a6c-c4c4b7f1f211', 'qw
0.7, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('71c78d46-962b-447e-945d-a3a3011add24', 'qwe
0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
INSERT INTO "public"."t_model_support" VALUES ('6020c67b-3057-40b0-986a-264f875ef2f6', 'en
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('e5f9369c-4fbf-4404-825e-de68bb894ace', 'gl
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('7eab132c-7c1f-41b3-8c3d-ac613d3bfdec', 'gl
1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性",
和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-04
INSERT INTO "public"."t_model_support" VALUES ('86c474d8-69fb-4d0b-8beb-66103dd38ce5', 'De
"min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('20141eae-a869-40eb-b77d-d8acff76e1e3', 'qw
```

```
"max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('fa49750c-2543-4454-bbc5-7109c0f076a2', 'er
0.1, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('1f10e545-19b2-4279-ae6d-0ef27d72c5a4', 'D
"min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('12b89432-2dab-4a40-8eef-e5dee4d409f1', 'D
1, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('ff4f800f-508f-427d-aa36-7d604e9a3995', 'D
1, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('05164a70-7f01-480c-8388-adc8b7a3116f', 'Do
1, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('38b2a229-4af6-4d02-8653-0f07bf22c0af', 'L
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('564726f7-dc4f-414f-8390-d758bfdb3aff', 'S
1, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
INSERT INTO "public"."t_model_support" VALUES ('172a1949-39a1-4aa8-a786-bd0b23386a8b', 'M
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.66613
INSERT INTO "public"."t_model_support" VALUES ('964ccafd-2809-4c14-858b-06f6b51bc9db', 'Ll
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('17fe5d5d-79f7-4500-9537-794c82014108', 'hu
"max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('3e8fa316-74b1-4c9e-aa68-408027179f1e', 'gp
"max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('8a10267b-0cf1-43e9-a2af-8b07f59773d2', 'g
"help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性", "key
temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('ff3f3772-937a-49fa-9f58-47e360deb8b2', 'gp
2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性",
和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-04
INSERT INTO "public"."t_model_support" VALUES ('929a2629-9710-4127-9bda-521b8fe794b7', 'g
"max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
```

```
271 而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
272 INSERT INTO "public"."t_model_support" VALUES ('5f97ea67-72b3-4533-a79f-56770c2bd3eb', 'q
273 "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
274 机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
275 INSERT INTO "public"."t_model_support" VALUES ('130ec082-4e3f-4358-bf62-e38714c5e462', 'qw
276 "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
277 机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
278 INSERT INTO "public"."t_model_support" VALUES ('9dcce7a7-4cdd-4846-b26a-f502eaa65f0e', 'Mo
279 "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
280 机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
281 INSERT INTO "public"."t_model_support" VALUES ('56866717-9181-4367-8341-3489f7745ef1', 'qw
282 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
283 一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
284 INSERT INTO "public"."t_model_support" VALUES ('82ac2479-2a2a-4540-875c-9c3dc15059e8', 'Mo
285 1, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散
286 随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40
287 INSERT INTO "public"."t_model_support" VALUES ('15a65488-7db0-4fbe-ba78-f0d3aab814fd', 'g
288 "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
289 而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
290 INSERT INTO "public"."t_model_support" VALUES ('82ac2479-2a2a-4540-875c-9c3dc1505919', 'de
291 "default": 1, "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数
292 越弱；数值越大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '20
293 INSERT INTO "public"."t_model_support" VALUES ('09b1bd04-e3c2-44d6-8f79-a5514114a8df', 'de
294 "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}]', 0, '202
295 INSERT INTO "public"."t_model_support" VALUES ('06b3d705-41f0-4181-b71b-d4ef6cb860d3', 'de
296 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}]', 0,
297 INSERT INTO "public"."t_model_support" VALUES ('6d0d4515-eab4-4927-8293-fad36d6c6375', 'hu
298 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
299 一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
300 INSERT INTO "public"."t_model_support" VALUES ('51d233e9-cc25-4856-96a0-243389ae1081', 'm
301 "min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
302 机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
303 INSERT INTO "public"."t_model_support" VALUES ('d95c30c2-5092-40e0-ad37-19a360bdca8f', 'mo
304 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
305 一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139
306 INSERT INTO "public"."t_model_support" VALUES ('8f4f0783-8f8f-4518-b080-cc67e803ded5', 'gp
307 "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性", "key
308 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-04 0
309 INSERT INTO "public"."t_model_support" VALUES ('db1f84d2-e37f-4731-8be0-23bbc4084394', 'gp
310 "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
311 机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
312 INSERT INTO "public"."t_model_support" VALUES ('e15e6c2c-959b-4e50-8748-a4f7979e46c0', 'g
313 "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
```

```
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('a6e9fe71-a462-483c-8502-2d44cf1f6dd2', 'gpt
1.0, "min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发
大，随机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 0
INSERT INTO "public"."t_model_support" VALUES ('ab347c18-3356-4c6a-9b4a-824a25cab617', 'gp
"min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('82ac2479-2a2a-4540-875c-9c3dc15059e1', 'qw
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 14:42:58', '202
INSERT INTO "public"."t_model_support" VALUES ('bcadc420-0d80-4777-b352-4152a4f8be4f', 'gp
"min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:
INSERT INTO "public"."t_model_support" VALUES ('35dee1b5-1b1c-41bd-8e83-c9ea41dba4af', 'qw
2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name": "随机性",
和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139', '2025-03-06
INSERT INTO "public"."t_model_support" VALUES ('c48d4efd-8695-4148-96be-c1cd3385bc56', 'g
"max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"name":
而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-04 03:40:13.666139',
INSERT INTO "public"."t_model_support" VALUES ('72ac2479-2a2a-4540-875c-9c3dc1515919', 'q
"min": 0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"
机性越强。一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-03-23 14:42:5
INSERT INTO "public"."t_model_support" VALUES ('82ac2479-2a2a-4540-875c-9c3dc1516919', 'Ca
0.0, "max": 1.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"}, {"nam
一般而言，top_p 和 temperature 两个参数选择一个进行调整即可。"}]', 0, '2025-04-22 14:42:58', '202
INSERT INTO "public"."t_model_support" VALUES ('fe239a60-de11-472f-81f0-3cbedbddcd9b', 'd
"min": 0.0, "max": 2.0, "help": "控制生成结果的多样性和随机性。数值越小，越严谨；数值越大，越发散。"

-- ----------------------------
-- Primary Key structure for table t_model_support
-- ----------------------------
ALTER TABLE "public"."t_model_support" ADD CONSTRAINT "t_model_support_pkey" PRIMARY KEY


/*
 Navicat Premium Data Transfer

 Source Server         : 192.168.7.2-platform
 Source Server Type    : PostgreSQL
 Source Server Version : 140005
 Source Host           : 192.168.7.2:5432
 Source Catalog        : platform
 Source Schema         : public
```

```
357
358  Target Server Type    : PostgreSQL
359  Target Server Version : 140005
360  File Encoding         : 65001
361
362  Date: 27/02/2025 17:40:04
363 */
364
365
366 -- ----------------------------
367 -- Table structure for t_auth_instance
368 -- ----------------------------
369 DROP TABLE IF EXISTS "public"."t_auth_instance";
370 CREATE TABLE "public"."t_auth_instance" (
371   "id" "pg_catalog"."uuid" NOT NULL,
372   "name" "pg_catalog"."varchar" COLLATE "pg_catalog"."default" NOT NULL,
373   "template_id" "pg_catalog"."varchar" COLLATE "pg_catalog"."default" NOT NULL,
374   "template_name" "pg_catalog"."varchar" COLLATE "pg_catalog"."default",
375   "template_description" "pg_catalog"."varchar" COLLATE "pg_catalog"."default",
376   "provider_code" "pg_catalog"."varchar" COLLATE "pg_catalog"."default" NOT NULL,
377   "type" "pg_catalog"."int4",
378   "content" "pg_catalog"."text" COLLATE "pg_catalog"."default",
379   "template_params" "pg_catalog"."text" COLLATE "pg_catalog"."default",
380   "status" "pg_catalog"."int4" NOT NULL,
381   "create_time" "pg_catalog"."timestamp" NOT NULL,
382   "update_time" "pg_catalog"."timestamp" NOT NULL
383 )
384 ;
385
386 -- ----------------------------
387 -- Primary Key structure for table t_auth_instance
388 -- ----------------------------
389 ALTER TABLE "public"."t_auth_instance" ADD CONSTRAINT "t_auth_instance_pkey" PRIMARY KEY
390
391
392 /*
393  Navicat Premium Data Transfer
394
395  Source Server         : 192.168.7.2-platform
396  Source Server Type    : PostgreSQL
397  Source Server Version : 140005
398  Source Host           : 192.168.7.2:5432
399  Source Catalog        : platform
```

```
 Source Schema        : public

 Target Server Type   : PostgreSQL
 Target Server Version : 140005
 File Encoding        : 65001

 Date: 27/02/2025 17:40:13
*/


-- ----------------------------
-- Table structure for t_auth_template
-- ----------------------------
DROP TABLE IF EXISTS "public"."t_auth_template";
CREATE TABLE "public"."t_auth_template" (
  "id" "pg_catalog"."uuid" NOT NULL,
  "name" "pg_catalog"."varchar" COLLATE "pg_catalog"."default" NOT NULL,
  "provider_code" "pg_catalog"."varchar" COLLATE "pg_catalog"."default",
  "type" "pg_catalog"."int4",
  "description" "pg_catalog"."varchar" COLLATE "pg_catalog"."default",
  "properties" "pg_catalog"."text" COLLATE "pg_catalog"."default",
  "status" "pg_catalog"."int4" NOT NULL DEFAULT 3,
  "create_time" "pg_catalog"."timestamp" NOT NULL,
  "update_time" "pg_catalog"."timestamp" NOT NULL
)
;

-- ----------------------------
-- Primary Key structure for table t_auth_template
-- ----------------------------
ALTER TABLE "public"."t_auth_template" ADD CONSTRAINT "t_auth_template_pkey" PRIMARY KEY


/*
 Navicat Premium Data Transfer

 Source Server        : 192.168.7.2-platform
 Source Server Type   : PostgreSQL
 Source Server Version : 140005
 Source Host          : 192.168.7.2:5432
 Source Catalog       : platform
 Source Schema        : public
```

```
 Target Server Type    : PostgreSQL
 Target Server Version : 140005
 File Encoding         : 65001

 Date: 01/03/2025 16:16:11
*/


-- ----------------------------
-- Table structure for agent_message
-- ----------------------------
DROP TABLE IF EXISTS "public"."agent_message";
CREATE TABLE "public"."agent_message" (
  "id" "pg_catalog"."uuid" NOT NULL,
  "conversation_id" "pg_catalog"."varchar" COLLATE "pg_catalog"."default",
  "message_id" "pg_catalog"."varchar" COLLATE "pg_catalog"."default" NOT NULL,
  "type" "pg_catalog"."int4",
  "status" "pg_catalog"."int4",
  "duration" "pg_catalog"."float4",
  "tokens" "pg_catalog"."int4",
  "result" "pg_catalog"."text" COLLATE "pg_catalog"."default",
  "timestamp" "pg_catalog"."int8" NOT NULL,
  "create_time" "pg_catalog"."timestamp" NOT NULL,
  "update_time" "pg_catalog"."timestamp" NOT NULL
)
;

-- ----------------------------
-- Indexes structure for table agent_message
-- ----------------------------
CREATE INDEX "agentmessagemodel_message_id" ON "public"."agent_message" USING btree (
  "message_id" COLLATE "pg_catalog"."default" "pg_catalog"."text_ops" ASC NULLS LAST
);

-- ----------------------------
-- Primary Key structure for table agent_message
-- ----------------------------
ALTER TABLE "public"."agent_message" ADD CONSTRAINT "agent_message_pkey" PRIMARY KEY ("id


INSERT INTO "public"."spliter" ("id", "spliter_id", "spliter", "create_time", "update_tim
INSERT INTO "public"."spliter" ("id", "spliter_id", "spliter", "create_time", "update_tim
INSERT INTO "public"."spliter" ("id", "spliter_id", "spliter", "create_time", "update_time
```

```
INSERT INTO "public"."t_auth_instance" ("id", "name", "template_id", "template_name", "te
('82b22f79-9695-417c-9e9c-f52259ba74c2', '汇智授权', '05b837b0-95ec-4983-ad5a-7014904b6536
'0be92a5cae3ee32e8560cc8ac5d272340c18edb19985dbd1f8a92355dd9108cb785842f04bf6990c2e20629c
'[{"name":"授权 key","key":"api_key","secret_type":0,"max_length":1000,"default_value":"",

-- ----------------------------
-- Add preset data for LLM node conversation isolation scenarios
-- ----------------------------
INSERT INTO "public"."agents" ("id", "name", "agent_uuid", "description", "avatar_image",
"is_knowledge", "knowledge_config", "entry_parameter", "auto_follow_up", "long_term_memor
"delisting_image", "websearch_config", "publish_time", "create_time", "update_time", "rele
't', '{}', NULL, 'f', 'f', 10, 6, 'f', '1', 'f', 1, NULL, NULL, NULL, '{}', NULL, '2025-0


CREATE VIEW "public"."v_application_basic" AS SELECT t.id,
   t.application_uuid,
   t.application_no,
   t.avatar_image,
   t.application_name,
   t.application_type,
   t.publish_status,
   t.description,
   t.delisting_cause,
   t.delisting_info,
   t.delisting_image,
   t.version,
   t.status,
   t.created_at,
   t.updated_at,
   t.published_at,
   t.created_by,
   t.updated_by,
   t.modified,
   t.update_version,
   t.privacy_status,
   t.release_status,
   t.score
  FROM ( SELECT d.id,
          d.uuid AS application_uuid,
          d.definition_no AS application_no,
          d.icon AS avatar_image,
          d.name AS application_name,
```

```
        d.application_type,
        d.publish_status,
        d.description,
        d.delisting_cause,
        d.delisting_info,
        d.delisting_image,
        d.version,
           CASE
              WHEN d.status = 1 THEN 2
              WHEN d.status = 2 THEN 1
              ELSE d.status
           END AS status,
        d.created_at,
        d.updated_at,
        d.published_at,
        d.created_by,
        d.updated_by,
        db.modified,
        db.update_version,
        db.privacy_status,
        '-1'::integer AS release_status,
        db.score
       FROM m_process_definition d
         JOIN m_process_definition_basic db ON d.definition_no::text = db.definition_no
      UNION ALL
       SELECT d.id,
        d.uuid AS application_uuid,
        d.definition_no AS application_no,
        d.icon AS avatar_image,
        d.name AS application_name,
        d.application_type,
        d.publish_status,
        d.description,
        d.delisting_cause,
        d.delisting_info,
        d.delisting_image,
        d.version,
           CASE
              WHEN d.status = 1 THEN 2
              WHEN d.status = 2 THEN 1
              ELSE d.status
           END AS status,
        d.created_at,
```

```
    d.updated_at,
    d.published_at,
    d.created_by,
    d.updated_by,
    db.modified,
    db.update_version,
    db.privacy_status,
    '-1'::integer AS release_status,
    db.score
   FROM a_process_definition d
     JOIN a_process_definition_basic db ON d.definition_no::text = db.definition_no
UNION ALL
 SELECT ag.id,
    ag.agent_uuid AS application_uuid,
    ag.agent_uuid AS application_no,
    ag.avatar_image,
    ag.name AS application_name,
    ag.agent_type AS application_type,
        CASE
            WHEN ag.agent_status = 1 THEN 1
            ELSE 0
        END AS publish_status,
    ag.description,
    ag.delisting_cause,
    ag.delisting_info,
    ag.delisting_image,
    ag.version::text AS version,
        CASE
            WHEN ag.is_delete = true THEN 3
            ELSE ag.agent_status
        END AS status,
    to_char(ag.create_time, 'YYYY-MM-DD HH24:MI:SS'::text) AS created_at,
    to_char(ag.update_time, 'YYYY-MM-DD HH24:MI:SS'::text) AS updated_at,
    to_char(ag.publish_time, 'YYYY-MM-DD HH24:MI:SS'::text) AS published_at,
    NULL::character varying AS created_by,
    NULL::character varying AS updated_by,
    ag.modified,
    NULL::bigint AS update_version,
        CASE
            WHEN ag.is_publicity THEN 1
            ELSE 0
        END AS privacy_status,
    ag.release_status,
```

```
        ag.score
    FROM agents ag) t;
```