Agent Runtime

CLI 使用指南

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

CLI 使用指南
agr CLI 安装与配置

最近更新时间：2026-05-27 20:38:30

`agr` 是 Agent Runtime 的命令行工具。本文说明如何安装 agr CLI、写入本地凭据，并验证配置是否可用。

前提条件

- 已获取腾讯云 `SecretId` 和 `SecretKey`。
- 安装环境可以使用 `go install`，或者可以从源码执行 `make build`。
- 如果运行环境的 Home 目录不可写，请提前准备一个可写配置路径，并在命令中通过 `--config` 显式传入。

步骤 1：安装 agr CLI

安装后建议立即验证版本，确认命令已加入 `PATH`。

方式一：使用 `go install`

```
go install github.com/TencentCloudAgentRuntime/agr-cli/cmd/agr@latest
agr version -o json
```

方式二：从源码构建

```
git clone https://github.com/TencentCloudAgentRuntime/agr-cli.git
cd agr-cli
make build
sudo cp agr /usr/local/bin/agr
agr version -o json
```

`agr version -o json` 返回 `Version`、`Commit` 和 `BuildTime` 字段，可用于确认当前安装版本。

步骤 2：写入本地凭据

`agr init` 只写本地 CLI 配置，不会创建远端资源。

```
export TENCENTCLOUD_SECRET_ID="<your-secret-id>"
export TENCENTCLOUD_SECRET_KEY="<your-secret-key>"

agr init \
```

```
  --secret-id "$TENCENTCLOUD_SECRET_ID" \
  --secret-key "$TENCENTCLOUD_SECRET_KEY" \
  --non-interactive
```

默认配置文件路径为 `~/.agr/config.toml` 。若 Home 目录不可写，可显式指定路径：

```
agr --config /path/to/agr-config.toml init \
  --secret-id "$TENCENTCLOUD_SECRET_ID" \
  --secret-key "$TENCENTCLOUD_SECRET_KEY" \
  --non-interactive
```

步骤 3：查看当前配置

完成初始化后，用 `agr status` 检查配置是否已被 CLI 识别。

```
agr status -o json
```

返回结果中重点确认以下字段：

- `Data.ConfigFound`
- `Data.ConfigLoaded`
- `Data.Auth.SecretId.Present`
- `Data.Auth.SecretKey.Present`
- `Data.Region`
- `Data.Domain`

如果使用了自定义配置路径，检查时需带上同一个 `--config` 参数。

步骤 4：诊断鉴权与连通性

当 `agr status` 已经能看到本地配置，但业务命令仍然失败时，使用 `agr doctor` 诊断。

```
agr doctor -o json
```

`agr doctor` 输出按检查项组织的结果，常见检查包括：

- `SecretId`
- `SecretKey`
- `TokenCache`
- `ConfigFilePermission`
- `Connectivity`

如果返回 `AUTH_FAILED` 或其他鉴权类错误，可继续执行：

```
agr explain AUTH_FAILED -o json
```

`agr explain` 返回错误含义、退出码、受影响命令以及建议修复动作。

可选：通过配置文件调整默认值

除命令行参数外，`agr` 支持从配置文件读取常用默认值。示例结构：

```
output = "json"
region = "ap-guangzhou"
domain = "tencentags.com"

[auth]
secret_id = "<your-secret-id>"
secret_key = "<your-secret-key>"

[sandbox]
default_user = "user"
```

在自动化脚本中，可以把 `output`、`region` 和 `domain` 写入配置文件以减少重复参数，再用 `agr status -o json` 复核实际生效值。

当前认证与配置入口

当前 `agr` CLI 未提供 `agr auth` 或 `agr auth whoami` 子命令。认证与本地配置请使用以下入口：

- `agr init`：首次写入本地凭据。
- `agr config path`：查看配置文件路径。
- `agr config show`：查看当前配置值及其来源。
- `agr config set`：更新配置文件中的默认值。
- `agr status`：检查配置是否被 CLI 识别。
- `agr doctor`：诊断鉴权与连通性。
- 全局参数：`--config`、`--region`、`--domain`、`--secret-id`、`--secret-key`

如果您在旧示例或其他材料中看到 `agr auth` 写法，请以当前 CLI 的 `agr --help`、`agr config --help` 和 `agr schema -o json` 输出为准。

验证结果

当以下条件全部满足时，表示安装与配置已完成：

- `agr version -o json` 能正常返回版本信息。
- `agr status -o json` 显示 `ConfigFound: true` 且凭据字段为 `Present: true`。
- `agr doctor -o json` 的 `Connectivity` 检查通过，或失败原因已明确指向待修复的凭据/网络问题。

清理或回滚

如果只是临时验证本地环境，完成后可删除测试配置并清理环境变量：

```
rm -f ~/.agr/config.toml
unset TENCENTCLOUD_SECRET_ID
unset TENCENTCLOUD_SECRET_KEY
```

如果使用了自定义配置路径，请将 `~/.agr/config.toml` 替换为实际文件路径。

agr CLI 命令参考

最近更新时间：2026-05-27 20:38:30

概述

`agr` 是 AGS 的命令行工具，用于管理 Tool、Instance、API Key、镜像预缓存任务，以及本地配置、诊断和原始 API 调用。
本文记录 `agr` 已公开的命令、参数形态和 JSON 输出约定。示例中的 `<tool-id>`、`<tool-name>`、`<instance-id>`、`<key-id>` 和镜像地址均为占位符。

使用前提

- 已安装 `agr`。
- 访问云端资源的命令需要可用的云端凭证。
- 自动化场景建议统一追加 `-o json --non-interactive`。

确认 CLI 可用：

```
agr version -o json
agr status -o json
agr doctor -o json
```

`agr version` 返回 `Data.Version`、`Data.Commit` 和 `Data.BuildTime`；`agr status` 返回 `Data.Auth`、`Data.Region`、`Data.Domain` 和 `Data.EffectiveOutput`；`agr doctor` 返回 `Data.Checks[]`，可直接查看凭证、连通性和本地配置状态。
如果需要初始化本地配置：

```
agr init --secret-id <your-secret-id> --secret-key <your-secret-key> --
overwrite -o json
```

成功时返回统一 JSON 信封，`Status` 为 `succeeded`，`Failure` 为 `null`。重新执行 `agr status -o json` 时，`Data.Auth.SecretId.Present` 和 `Data.Auth.SecretKey.Present` 应为 `true`。

全局参数与输出

全局参数

参数	含义
`-o, --output`	输出格式，支持 `text`、`json`、`ndjson`。

`--jq`	对 JSON 输出执行 `jq` 表达式提取。
`--non-interactive`	关闭交互行为。
`--config`	指定配置文件路径。
`--region`	指定访问地域，默认 `ap-guangzhou` 。
`--cloud-endpoint`	指定控制面 API Endpoint。
`--domain`	指定数据面基础域名，默认 `tencentags.com` 。
`--secret-id`	显式传入 `SecretId` 。
`--secret-key`	显式传入 `SecretKey` 。
`--debug`	输出调试信息到标准错误。
`--generate-skeleton`	为支持 request 的命令输出请求骨架。
`--no-browser`	关闭自动打开浏览器。
`--no-color`	关闭 ANSI 颜色输出。
`--internal`	内部 Endpoint 快捷参数，已标记为 deprecated。

JSON 输出信封

支持 JSON 输出的命令统一返回以下结构：

```
{
  "SchemaVersion": "agr.v1",
  "Command": "version",
  "Status": "succeeded",
  "Data": {},
  "Failure": null,
  "Warnings": [],
  "Meta": {
    "Backend": "cloud",
    "DurationMs": 0
  }
}
```

字段	含义
`SchemaVersion`	返回结构版本。
`Command`	实际执行的命令名。
`Status`	执行状态，常见值为 `succeeded` 或 `failed`。
`Data`	命令主体结果。
`Failure`	失败详情；成功时为 `null`。
`Warnings`	警告列表。
`Meta`	元数据，常见字段为 `Backend` 和 `DurationMs`。

顶层命令

命令	别名	说明	JSON 输出	备注
`agr tool`	`t`	管理 Tool 模板	支持	远端资源命令
`agr instance`	`i`	管理 Instance 和实例内操作	支持	`login`、`proxy` 为交互式
`agr apikey`	`ak`、`key`	管理 API Key	支持	远端资源命令
`agr image`	无	管理镜像相关任务	支持	公开子命令为 `precache`
`agr api`	无	发送原始 Cloud API 请求	支持	低层 escape hatch
`agr init`	无	初始化本地 CLI 配置	支持	本地配置命令
`agr explain`	无	解释错误码和退出码	支持	只读
`agr status`	无	查看当前 CLI 配置状态	支持	只读
`agr schema`	无	输出机器可读命令 schema	支持	只读
`agr doctor`	无	诊断本地配置和连通性	支持	只读

`agr version`	无	查看 CLI 版本	支持	只读
`agr completion`	无	生成 Shell 自动补全脚本	不支持	本地辅助命令

Tool 命令

`agr tool` 用于管理沙箱 Tool 模板。

命令	别名	说明	JSON 输出
`agr tool create`	无	创建 Tool	支持
`agr tool list`	无	列出 Tool	支持
`agr tool get <tool-id>`	无	查询 Tool 详情	支持
`agr tool update <tool-id>`	无	更新 Tool	支持
`agr tool delete <tool-id>`	无	删除 Tool	支持

创建 Tool

```
agr tool create -n <tool-name> -t <tool-type> --network SANDBOX -o json
```

常用参数：

参数	必填	含义
`-n, --name`	是	Tool 名称
`-t, --type`	是	Tool 类型
`--network`	否	网络模式，支持 `PUBLIC`、`VPC`、`SANDBOX`
`-d, --description`	否	Tool 描述
`--timeout`	否	默认超时，例如 `5m`、`300s`、`1h`
`--persistent`	否	是否创建持久化沙箱

`--mount`	否	存储挂载 DSL
`--tag`	否	标签，格式为 `key=value`
`--custom-configuration`	否	`CustomConfiguration` JSON、`@file` 或标准输入
`--log-configuration`	否	`LogConfiguration` JSON、`@file` 或标准输入
`--client-token`	否	幂等令牌
`--request`	否	完整请求体 JSON、`@file` 或标准输入

查看请求骨架：

```
agr tool create --generate-skeleton
```

输出的骨架包含 `NetworkConfiguration`、`StorageMounts`、`Tags` 等字段，可作为完整请求体的编辑起点。

查询、更新和删除 Tool

```
agr tool list --limit 20 -o json
agr tool get <tool-id> -o json
agr tool update <tool-id> --description "updated description" -o json
agr tool delete <tool-id> -o json
```

以上命令均支持 `-o json`，返回统一 JSON 信封。可通过 `agr tool <subcommand> --help` 查看各子命令的完整参数和过滤项。

Instance 命令

`agr instance` 用于管理基于 Tool 创建的运行实例，以及在实例内执行代码、Shell、文件传输、Browser 和 Mobile 操作。

生命周期命令

命令	别名	说明	JSON 输出
`agr instance create`	`c`	创建实例	支持
`agr instance list`	无	列出实例	支持

`agr instance get <instance-id>`	无	查询实例详情	支持
`agr instance update <instance-id>`	无	更新实例	支持
`agr instance pause <instance-id>`	无	暂停实例	支持
`agr instance resume <instance-id>`	无	恢复实例	支持
`agr instance delete <instance-id>`	无	删除实例	支持

创建实例

```
agr instance create --tool-name <tool-name> -o json
```

或通过 Tool ID：

```
agr instance create --tool-id <tool-id> -o json
```

常用参数：

参数	必填	含义
`-t, --tool-name`	与 `--tool-id` 二选一	Tool 名称
`--tool-id`	与 `--tool-name` 二选一	Tool ID
`--timeout`	否	实例超时，例如 `5m` 、 `300s` 、 `1h` ；裸数字按秒处理
`--auth-mode`	否	认证模式，支持 `DEFAULT` 、 `TOKEN` 、 `NONE` 、 `PUBLIC`
`--mount-option`	否	挂载覆盖项，可重复传入
`--metadata`	否	JSON 数组形式的元数据
`--metadata-kv`	否	`KEY=VALUE` 形式的元数据
`--custom-configuration`	否	`CustomConfiguration` JSON、 `@file` 或标准输入

`--client-token`	否	幂等令牌
`--request`	否	完整请求体 JSON、`@file` 或标准输入

查看请求骨架：

```
agr instance create --generate-skeleton
```

输出的骨架包含 `ToolId`、`ToolName`、`Timeout`、`MountOptions` 和 `Metadata` 等字段。

查询与状态过滤

```
agr instance list --limit 20 -o json
agr instance list --status RUNNING -o json
agr instance get <instance-id> -o json
```

`agr instance list` 支持的 `--status` 过滤值：

状态值	含义
`STARTING`	正在启动
`RUNNING`	运行中
`FAILED`	启动失败
`STOPPING`	正在停止
`STOPPED`	已停止

更新、暂停、恢复和删除

```
agr instance update <instance-id> --timeout 10m -o json
agr instance pause <instance-id> -o json
agr instance resume <instance-id> -o json
agr instance delete <instance-id> --ignore-not-found -o json
```

删除场景建议保留 `--ignore-not-found`，便于脚本幂等清理。

实例内代码执行和 Shell 执行

命令	JSON 输出	NDJSON 输出	备注
`agr instance code run <instance-id>`	支持	支持	`--stream` 适合配合 `-o ndjson`
`agr instance exec <instance-id> -- <command>`	支持	支持	远端命令必须放在 `--` 后

代码执行示例：

```
agr instance code run <instance-id> -l python -c "print('hello')" -o
json
```

Shell 执行示例：

```
agr instance exec <instance-id> -o json --cwd /workspace --env
APP_ENV=prod -- python app.py
```

需要实时输出时，使用 `--stream -o ndjson` 或默认文本输出。`--stream` 不适合与 `-o json` 同时使用。
临时实例：追加 `--create-temp-instance`（默认 `--cleanup always`），命令完成后自动删除临时实例。
如果改为 `--cleanup never`，需手动执行 `agr instance delete`。
支持的 `code run` 语言：

- `python`
- `javascript`
- `typescript`
- `r`
- `java`
- `bash`

文件传输

```
agr instance file upload <instance-id> ./app.py /workspace/app.py -o
json
agr instance file download <instance-id> /workspace/result.json
./result.json -o json
```

两条命令均支持 `-o json` 和 `--user` 参数。

Browser、Mobile 和交互式入口

命令	JSON 输出	交互式	说明
`agr instance browser vnc <instance-id>`	支持	否	返回 Browser 实例的 VNC URL
`agr instance mobile connect <instance-id>`	支持	否	建立后台隧道并执行 `adb connect`
`agr instance mobile disconnect [instance-id]`	支持	否	支持 `--all`
`agr instance mobile list`	支持	否	列出当前活动连接
`agr instance mobile adb <instance-id> -- <adb-args...>`	支持	否	执行 `adb` 命令
`agr instance proxy <instance-id> [local_port:]<remote_port>`	不支持	是	本地端口转发
`agr instance login <instance-id>`	不支持	是	当前终端内直接打开 PTY 会话

示例：

```
agr instance browser vnc <instance-id> -o json
agr instance mobile list -o json
agr instance mobile adb <instance-id> -o json -- shell ls /sdcard
agr instance proxy <instance-id> 3000:8080 --address 127.0.0.1
agr instance login <instance-id> --user root
```

`browser vnc`、`mobile list` 和 `mobile adb` 支持 JSON 输出；`proxy` 和 `login` 为交互式命令，不返回 JSON。

API Key 命令

`agr apikey` 用于管理 API Key。

```
agr apikey create -n <name> -o json
agr apikey list -o json
```

```
agr apikey delete <key-id> -o json
```

`create` 、 `list` 和 `delete` 均支持 `-o json` ，返回统一 JSON 信封。

镜像预缓存命令

`agr image` 公开的命令组为 `precache` 。

```
agr image precache create --image <image-ref> --image-registry-type
enterprise -o json
agr image precache get <image-digest> --image <image-ref> --image-
registry-type enterprise -o json
```

查看请求骨架：

```
agr image precache create --generate-skeleton
```

输出包含 `Image` 和 `ImageRegistryType` 两个请求字段。 `--image-registry-type` 支持 `enterprise` 和 `personal` 。

原始 API 调用

`agr api` 只公开 `call` 子命令，用于发送原始 Cloud API 请求。常规资源管理优先使用 `tool` 、 `instanc e` 、 `apikey` 和 `image` 命令组。

```
agr api call --help
```

该命令组属于低层接口，适合 typed surface 尚未覆盖的场景。

常见报错与排查

凭证未配置

症状：远端资源命令返回 `Failure.Code: "MISSING_CLOUD_CREDENTIALS"` 。

```
{
  "SchemaVersion": "agr.v1",
  "Command": "tool.list",
  "Status": "failed",
  "Data": null,
  "Failure": {
```

```
    "Code": "MISSING_CLOUD_CREDENTIALS",
    "Kind": "auth",
    "Message": "cloud API credentials are required",
    "Hint": "Run: agr init --secret-id <id> --secret-key <key>, or set
TENCENTCLOUD_SECRET_ID and TENCENTCLOUD_SECRET_KEY.",
    "Retryable": false
  }
}
```

处理方式：

- **运行** `agr init --secret-id <your-secret-id> --secret-key <your-secret-key> --overwrite -o json` **。**
- **或在当前 shell 注入** `TENCENTCLOUD_SECRET_ID` **和** `TENCENTCLOUD_SECRET_KEY` **后重试。**
- **用** `agr status -o json` **和** `agr doctor -o json` **复核状态。**

网络不通或 Endpoint 配置错误

症状：命令返回 `Failure.Code: "ClientError.NetworkError"` **。**

```
{
  "SchemaVersion": "agr.v1",
  "Command": "tool.list",
  "Status": "failed",
  "Data": null,
  "Failure": {
    "Code": "ClientError.NetworkError",
    "Kind": "error",
    "Message": "Fail to get response because Post
\"https://127.0.0.1:9/\": dial tcp 127.0.0.1:9: connect: connection
refused",
    "Hint": "Run 'agr doctor' to diagnose configuration and
connectivity.",
    "Retryable": false
  }
}
```

处理方式：

- **先运行** `agr doctor -o json` **，确认** `Connectivity` **检查项是否为** `ok` **。**
- **检查** `--cloud-endpoint` **、代理和网络连通性配置。**

命令不存在或参数不匹配

症状：CLI 返回 `INVALID_USAGE`，或提示 `unknown command`。

```
Error: unknown command "sandbox" for "agr" (INVALID_USAGE)
Hint: Run 'agr --help' or 'agr schema -o json' to inspect valid commands
and flags.
```

处理方式：

- **用 `agr --help`、`agr <command> --help` 或 `agr schema -o json` 复核当前版本支持的命令和参数。**
- **用 `agr version -o json` 核对当前 CLI 版本。**

本地缓存目录不可写

症状：`agr doctor -o json` 的 `Data.Checks[]` 中出现 `TokenCache` 警告。
处理方式：按提示创建本地目录并收紧权限，例如 `mkdir -p ~/.agr && chmod 700 ~/.agr`。

清理试用资源

如果试用过程中创建了远端资源，结束后请及时清理：

```
agr instance delete <instance-id> --ignore-not-found -o json
agr tool delete <tool-id> -o json
agr apikey delete <key-id> -o json
```

如果使用了 `--create-temp-instance`（默认 `--cleanup always`），临时实例会自动删除；如果改为 `--cleanup never`，请手动执行 `agr instance delete`。

已知限制

- **`agr completion` 只生成补全脚本，不返回 JSON。**
- **`agr instance login` 和 `agr instance proxy` 为交互式命令，不支持 JSON 输出。**
- **`agr instance code run --stream` 更适合与 `-o ndjson` 或默认文本输出搭配。**
- **`agr --help` 未公开 `snapshot`、`sandbox` 或 `docs` 顶层命令；不要把这些命令写成已发布 CLI 能力。**

agr CLI 常用操作示例

最近更新时间：2026-05-27 20:38:30

本页汇总 `agr` CLI 的常用操作示例，覆盖环境检查、Tool 查询、Instance 生命周期管理、代码执行、Shell 命令、文件传输、端口代理和故障排查。
本页示例假设您已完成 CLI 安装和凭证配置，相关步骤请参见 安装与配置。
除交互式前台命令外，示例默认使用 `-o json`，便于脚本集成和配合 `--jq` 提取字段；自动化场景建议同时追加 `--non-interactive`。需要实时流式输出时，可切换到 `-o ndjson`；`agr instance proxy` 为前台交互式命令，不使用 JSON 输出。

使用前检查

查看 CLI 版本

```
agr version -o json
```

返回结果中包含版本号和构建信息：

```
{
  "SchemaVersion": "agr.v1",
  "Command": "version",
  "Status": "succeeded",
  "Data": {
    "Version": "51d952b",
    "Commit": "51d952b",
    "BuildTime": "2026-05-24_11:23:07"
  }
}
```

检查当前配置与凭证状态

```
agr status -o json
```

重点确认以下字段：

- `Data.Auth.SecretId.Present`
- `Data.Auth.SecretKey.Present`
- `Data.Region`
- `Data.Domain`

- `Data.EffectiveOutput`

凭证已生效时，返回结果类似：

```
{
  "SchemaVersion": "agr.v1",
  "Command": "status",
  "Status": "succeeded",
  "Data": {
    "Auth": {
      "SecretId": {
        "Present": true,
        "Source": "TENCENTCLOUD_SECRET_ID"
      },
      "SecretKey": {
        "Present": true,
        "Source": "TENCENTCLOUD_SECRET_KEY"
      }
    },
    "ConfigFound": false,
    "ConfigLoaded": false,
    "Region": "ap-guangzhou",
    "Domain": "tencentags.com",
    "EffectiveOutput": "json"
  }
}
```

诊断本地配置与连通性

```
agr doctor -o json
```

重点确认 `Data.Checks[]` 中的 `SecretId`、`SecretKey` 和 `Connectivity` 检查项。连通性正常时输出类似：

```
{
  "SchemaVersion": "agr.v1",
  "Command": "doctor",
  "Status": "succeeded",
  "Data": {
    "Checks": [
```

```
      {
        "Name": "Connectivity",
        "Status": "ok",
        "Message": "API reachable, credentials valid"
      }
    ]
  }
}
```

查询 Tool 和 Instance

查看 Tool 列表

```
agr tool list --limit 20 -o json --non-interactive
```

返回结果包含 `Data.Items[]` 和分页信息：

```
{
  "SchemaVersion": "agr.v1",
  "Command": "tool.list",
  "Status": "succeeded",
  "Data": {
    "Items": [
      {
        "ToolId": "sdt-xxxxxxxx",
        "ToolName": "example-tool",
        "ToolType": "android-world",
        "Status": "ACTIVE",
        "NetworkConfiguration": {
          "NetworkMode": "PUBLIC"
        }
      }
    ],
    "Pagination": {
      "Limit": 20,
      "Offset": 0,
      "Total": 487
    }
  }
```

```
}
```

查看 Tool 详情

```
agr tool get <tool-id> -o json --non-interactive
```

输出中包含目标 Tool 的 `ToolId`、`ToolName`、`ToolType`、`Status` 和 `NetworkConfiguration`。如果只需确认 Tool 是否可用，优先检查 `Data.Status` 是否为 `ACTIVE`。

查看实例列表

```
agr instance list --limit 20 -o json --non-interactive
```

返回结果包含 `Data.Items[]` 和分页信息：

```
{
  "SchemaVersion": "agr.v1",
  "Command": "instance.list",
  "Status": "succeeded",
  "Data": {
    "Items": [
      {
        "InstanceId": "f8e694...",
        "ToolId": "sdt-xxxxxxxx",
        "ToolName": "example-tool",
        "Status": "STOPPED",
        "AuthMode": "DEFAULT",
        "NetworkMode": "PUBLIC"
      }
    ],
    "Pagination": {
      "Limit": 20,
      "Offset": 0,
      "Total": 42370
    }
  }
}
```

如果只关心运行中的实例，可追加 `--filters '[{"Name":"Status","Values":["RUNNING"]}]'`。

查看实例详情

```
agr instance get <instance-id> -o json --non-interactive
```

输出包含 `InstanceId`、`ToolId`、`ToolName`、`Status`、`TimeoutSeconds` 和 `UpdateTime`：

```
{
  "SchemaVersion": "agr.v1",
  "Command": "instance.get",
  "Status": "succeeded",
  "Data": {
    "InstanceId": "f8e694...",
    "ToolId": "sdt-xxxxxxxx",
    "ToolName": "example-tool",
    "Status": "STOPPED",
    "AuthMode": "DEFAULT",
    "NetworkMode": "PUBLIC",
    "TimeoutSeconds": 0
  }
}
```

创建和管理 Instance

创建实例并提取 `InstanceId`

如需先查看完整响应，可直接执行：

```
agr instance create --tool-id <tool-id> --timeout 5m -o json
```

自动化脚本通常再配合 `--jq` 提取 `InstanceId`：

```
INSTANCE_ID=$(agr instance create --tool-id <tool-id> --timeout 5m -o
json --jq '.Data.InstanceId')
```

成功信号：

- 命令返回统一 JSON 信封，`Command` 为 `instance.create`。
- `Data.InstanceId` 为非空字符串。

- 后续执行 `agr instance get "$INSTANCE_ID" -o json --non-interactive` 时，可以查询到该实例。

如果更习惯按 Tool 名称创建，也可以使用：

```
agr instance create --tool-name <tool-name> --timeout 5m -o json
```

查询实例状态

```
agr instance get "$INSTANCE_ID" -o json --non-interactive
```

重点检查 `Data.Status`。当前常见状态包括：

- `STARTING`
- `RUNNING`
- `FAILED`
- `STOPPING`
- `STOPPED`

如果实例刚创建完成但仍为 `STARTING`，请等待片刻后重新查询，直到状态进入 `RUNNING` 再继续执行实例内操作。

暂停和恢复实例

```
agr instance pause "$INSTANCE_ID" -o json
agr instance get "$INSTANCE_ID" -o json --non-interactive

agr instance resume "$INSTANCE_ID" -o json
agr instance get "$INSTANCE_ID" -o json --non-interactive
```

执行后再用 `agr instance get "$INSTANCE_ID" -o json --non-interactive` 观察 `Data.Status` 是否切换为期望状态。如果状态未变化，请稍后重试，并确认实例当前生命周期是否允许暂停或恢复。

删除实例

```
agr instance delete "$INSTANCE_ID" --ignore-not-found -o json
```

成功信号：

- 命令返回统一 JSON 信封，`Status` 为 `succeeded`。
- `Data.Deleted` 为 `1`，`Data.Failed` 为 `0`。

- 如需补充验证，可重新执行 `agr instance list --limit 20 -o json --non-interactive`，确认该实例不再出现在当前列表结果中。

在实例内执行代码和命令

在已有实例中执行代码

```
agr instance code run "$INSTANCE_ID" -l python -c 'print("hello from
agr")' -o json
```

执行结果默认以 JSON 返回；如果需要实时查看 stdout/stderr，请改用 `--stream -o ndjson`。
从文件读取代码：

```
agr instance code run "$INSTANCE_ID" -l python -f ./hello.py -o json
```

如果需要实时输出，请改用：

```
agr instance code run "$INSTANCE_ID" -l python -c 'print("hello from
agr")' --stream -o ndjson
```

`--stream` 不能与 `-o json` 同时使用。

使用临时实例执行一次性代码

```
agr instance code run --create-temp-instance -t <tool-name> -l python -c
'print("hello from temp instance")' -o json
```

适用场景：

- 只想验证 Tool 是否可运行。
- 不希望手动管理实例生命周期。

如果要保留临时实例便于继续排查，可追加 `--cleanup never`，然后手动执行 `agr instance delete`。

执行 Shell 命令

```
agr instance exec "$INSTANCE_ID" -o json -- pwd
```

指定工作目录和环境变量：

```
agr instance exec "$INSTANCE_ID" -o json --cwd /workspace --env
APP_ENV=dev -- bash -lc 'echo $APP_ENV && pwd'
```

远端命令必须放在 `--` 之后，否则 CLI 会把后续参数继续解析为本地 flag。
如果命令返回 `INTERNAL_ERROR`，请先确认实例处于 `RUNNING`，再结合 `agr doctor -o json` 和 `agr instance get "$INSTANCE_ID" -o json --non-interactive` 排查实例健康状态。

文件传输和端口代理

上传文件到实例

```
agr instance file upload "$INSTANCE_ID" ./notes.txt /home/user/notes.txt
-o json --non-interactive
```

上传后可通过后续业务命令读取目标路径，或在应用内直接验证该文件是否已经可用。如果命令返回 `INTERNAL_ERROR`，请先确认实例处于 `RUNNING`，并检查目标路径、实例内用户权限和当前实例类型是否支持该文件操作。

从实例下载文件

```
agr instance file download "$INSTANCE_ID" /home/user/notes.txt
./notes.txt -o json --non-interactive
```

下载后先在本地检查目标文件是否已经生成；如需进一步确认，可将本地文件内容与远端源文件再次比对。如果命令返回 `INTERNAL_ERROR`，请先确认实例状态、源路径和当前实例类型是否支持该文件操作。

建立端口代理

```
agr instance proxy "$INSTANCE_ID" 3000:8080
```

绑定所有本地网卡：

```
agr instance proxy "$INSTANCE_ID" 8080 --address 0.0.0.0
```

`agr instance proxy` 以前台方式运行，建立本地端口与实例端口的映射。执行后直接访问本地映射端口验证实例内服务是否可达；停止代理时终止当前进程即可。

获取 Browser 实例的 VNC 地址

```
agr instance browser vnc <instance-id> -o json
```

如需指定 VNC 端口：

```
agr instance browser vnc <instance-id> --port 9000 -o json
```

返回结果中包含 `Data.InstanceId`、`Data.VncUrl` 和 `Data.CdpUrl`。这些 URL 会附带临时访问参数，请勿直接写入脚本仓库或长期日志。

故障排查

云端凭证未配置

症状：

- `agr status -o json` 中 `Data.Auth.SecretId.Present` 或 `Data.Auth.SecretKey.Present` 为 `false`。
- `agr doctor -o json` 中 `SecretId`、`SecretKey` 检查项为 `warning`。
- 查询或管理命令返回认证相关错误，例如 `MISSING_CLOUD_CREDENTIALS`。

处理方式：

```
agr init --secret-id <your-secret-id> --secret-key <your-secret-key> --
non-interactive
agr status -o json
agr doctor -o json
```

网络不通或 Endpoint 配置错误

症状：

- 已配置凭证后，`agr doctor -o json` 中 `Connectivity` 检查项不是 `ok`。
- 业务命令返回网络连接、TLS 或超时类错误。

处理方式：

- 先运行 `agr doctor -o json`，确认问题是否出在连通性。
- 如果仍处于未配置凭证状态，先完成初始化，再重新执行连通性检查。
- 检查 `--cloud-endpoint`、`--region`、代理设置和本地网络出口。
- 如果只想恢复默认设置，可先移除自定义的 `--cloud-endpoint` 参数后重试。

命令或参数与当前版本不匹配

症状：

- CLI 返回 `unknown command`、`INVALID_USAGE` 或参数解析错误。
- 命令以退出码 `2` 结束，表示用法错误。

处理方式：

```
agr --help
agr <command> --help
agr schema -o json
agr version -o json
agr explain exit-codes -o json
```

`agr explain exit-codes -o json` 可用于查看当前 CLI 的退出码含义：

```
{
  "0": "success",
  "1": "error",
  "2": "usage",
  "4": "auth",
  "255": "remote_execution_failed"
}
```

如果旧脚本仍然使用已经不存在的命令组，请以当前版本的 `agr --help` 和 `agr schema -o json` 为准进行调整。

本地缓存目录不可写

症状：

- `agr doctor -o json` 的 `Data.Checks[]` 中出现 `TokenCache` 警告。

处理方式：

- 为 `~/.agr` 准备可写目录，或在命令中使用 `--config` 指向可写配置文件路径。

相关文档

- 安装与配置
- agr CLI 命令参考
- 启动 Sandbox Instance
- 查询 Sandbox Instance
- 停止与超时
- 常见错误码