ALIBABA CLOUD

阿里云

CDN

边缘程序

文档版本：20200912

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.功能介绍 ---------- 05
2.EdgeRoutine场景和代码示例 ---------- 07
3.EdgeRoutine CLI工具使用说明 ---------- 33

1.功能介绍

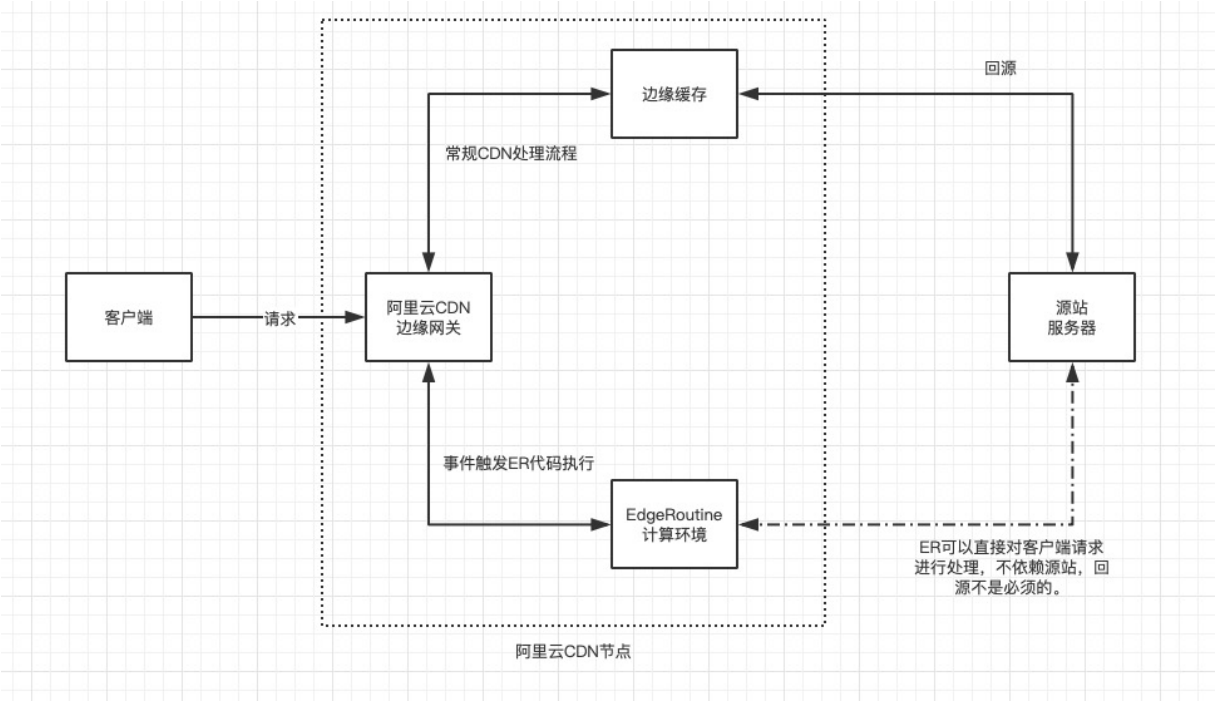
EdgeRoutine（边缘程序，以下简称ER）是阿里云CDN推出的边缘Serverless计算环境。ER目前针对CDN场景，支持在CDN边缘节点执行您自行编写的JavaScript代码。您无需关心代码部署的机器硬件配置、部署region、调度和弹性扩容。ER代码一经上传，即可完成全球阿里云CDN节点的部署，在全球边缘执行，根据您的代码逻辑个性化处理全球各地的请求。

为什么使用EdgeRoutine？

传统的CDN服务是纯粹的缓存和分发服务，缺乏可以直接提供给您的计算能力。访问CDN的海量请求中，复杂的计算逻辑必须回服务器源站执行，这增加了您的服务器消耗以及架构的复杂性。ER可提供直接在CDN边缘节点计算处理的能力，将极大提高CDN的可定制化，可编程化，从而大量减少需回源的请求，降低用户的请求延时。同时CDN边缘节点拥有天然的高可用、高伸缩、全球负载均衡的特性，边缘的计算服务可应用于更多的使用场景。阿里云CDN的ER可为您提供边缘节点的计算环境，为您的代码部署提供更多的选择。

原理

ER直接运行在阿里云CDN边缘节点上，提供独立的JS运行环境，供您部署JS代码。



应用场景

由于ER内部有完整的JS运行环境，同时给予您完全的请求控制，HTTP/HTTPS异步自请求的能力，可以使ER能够覆盖较多的应用场景：

应用场景	可用于...
鉴权	CDN自定义鉴权，异步鉴权
Originless （无源站）	静态页面

应用场景	可用于...
网络安全	• 反爬虫 • 边缘WAF
日志	• 自动生成边缘日志 • 记录回源时间
DevOps	• AB测试 • 设置GitLab、GitHub、Jenkins的webhook • 编写边缘IM机器人，例如：slack机器人

API Gateway	• 边缘GraphQL
	• 边缘网关
	  ◦ 限流
	  ◦ API参数验证

CDN	• 边缘Cache预热
	• 边缘内容生成
	• 边缘请求回多源站
	  ◦ 合并多源站内容
	  ◦ 选择最快的源站响应
	• 边缘动态回源，自定义回源算法

Web/小程序	• 边缘渲染
	  ◦ Markdown渲染
	  ◦ ESI渲染
	  ◦ Nunjunk渲染
	• 边缘SSR
	  ◦ Vue.js
	  ◦ React.js
	• 页面延时优化、预热
	• 边缘杂项服务，例如：二维码生成

Beacon	边缘打点数据的捕获与分析

使用ER

目前ER处于邀测阶段，需要您提交工单申请ER使用权，审批（预计3个工作日）通过后，您可以通过CLI工具提交JS代码至阿里云CDN节点。

2.EdgeRoutine场景和代码示例

本文为您介绍阿里云边缘Serverless产品—EdgeRoutine 的简单场景示例和测试说明。Edgeroutine满足您在阿里云边缘节点运行代码的需求，提供轻量级可编程环境，利用阿里云CDN的2500+全球边缘节点，实现自动部署、就近接入的可编程CDN和Serverless服务。

CLI环境准备

本地环境准备请参见·EdgeRoutine CLI工具使用说明。

事件说明

您需要调用addEventListener函数去注册一个事件回调，目前支持fetch事件，fetch事件是由阿里云CDN的HTTP请求触发，即每次客户端访问CDN域名，将由CDN完成边缘就近接入，在边缘节点自动关联边缘Serverless服务，可以耦合到您的CDN业务生命周期（拦截或旁路），满足您的CDN可编程的需求，也可独立作为Serverless代替源站服务。

在fetch事件回调函数中，您必须使用event.respondWith去注册一个异步函数，该异步函数将返回一个Promise对象，Promise在JavaScript中是ES6异步的核心，可以理解为，在将来这个对象会被解析成真正的响应内容返回给CDN甚至客户端。几乎所有的程序的addEventListener都是如此调用。

示例如下：

```
async function handleRespond(event) {
  return fetch("http://www.example.com");
}

addEventListener('fetch', (event) => {
  event.respondWith(handleRespond(event));
});
```

Edgeroutine支持Web标准API - Service Worker API，兼容标准ES6语法，且大量的Node.js第三方库直接使用，也支持标准的JS开发模型，为了满足您的测试需求，我们整理了相关的若干示例，您可以直接测试。

我们提供 edge.ialicdn.com 域名用来测试已默认配置的的 JavaScript Demo，示例源码见文章末尾。

- 示例代码使用js构建不同代码片段用于解决各种常见场景，目前是15+场景供您参考和学习。
- 为区分不同场景的示例，须在请求的body中携带json格式的kv对来将请求路由到不同的函数。

Hello World

需求：该示例场景实现一个简单的边缘Serverless服务，无需回源，直接在边缘节点生成内容。

命令：

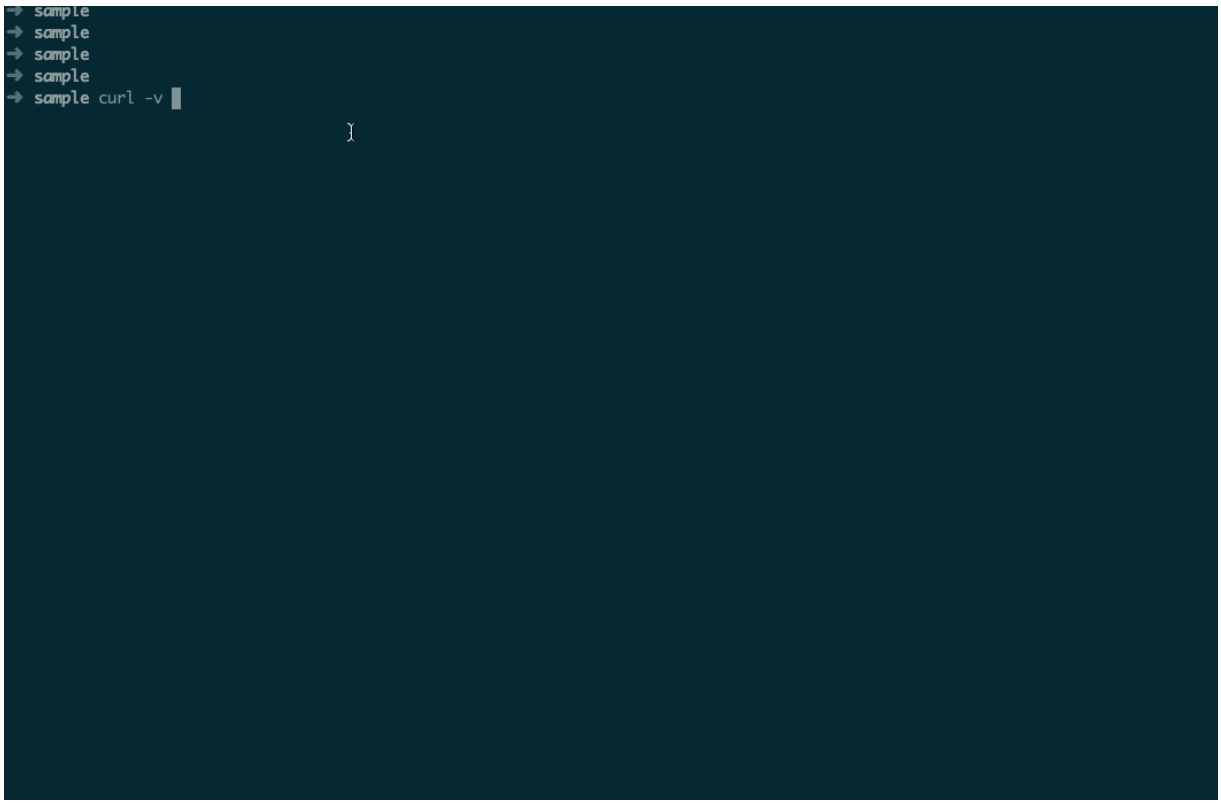
```
curl -v 'http://edge.ialicdn.com/a/b?x=y' -d '{"name": "helloworld"}'
```

Geo

需求：该示例场景实现一个简单的边缘打点服务，可以采集到边缘节点的请求相关信息：如IP、地理、设备信息等。

命令：

```
curl -v 'http://edge.ialicdn.com/a/b?x=y' -d '{"name": "geo"}' -H "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.80 Safari/537.36"
```

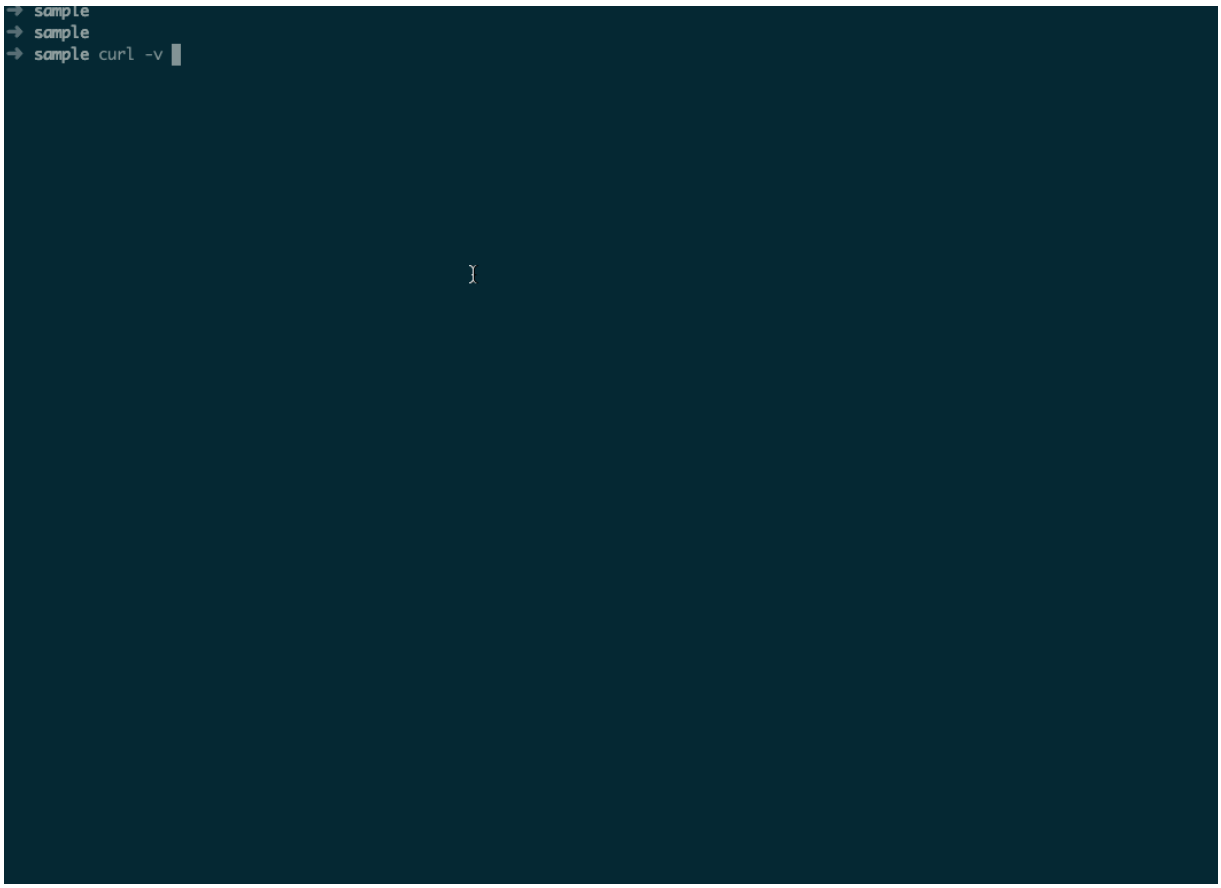


Fetch

需求：该示例场景实现一个简单的边缘代理服务，在JS代码中调用内置api fetch做了http自请求，响应给客户端fetch的最终内容。

命令：

```
curl -v 'http://edge.ialicdn.com/a/b?x=y' -d '{"name": "fetch", "url": "http://a.hongxiaolong.com/xx"}'
```

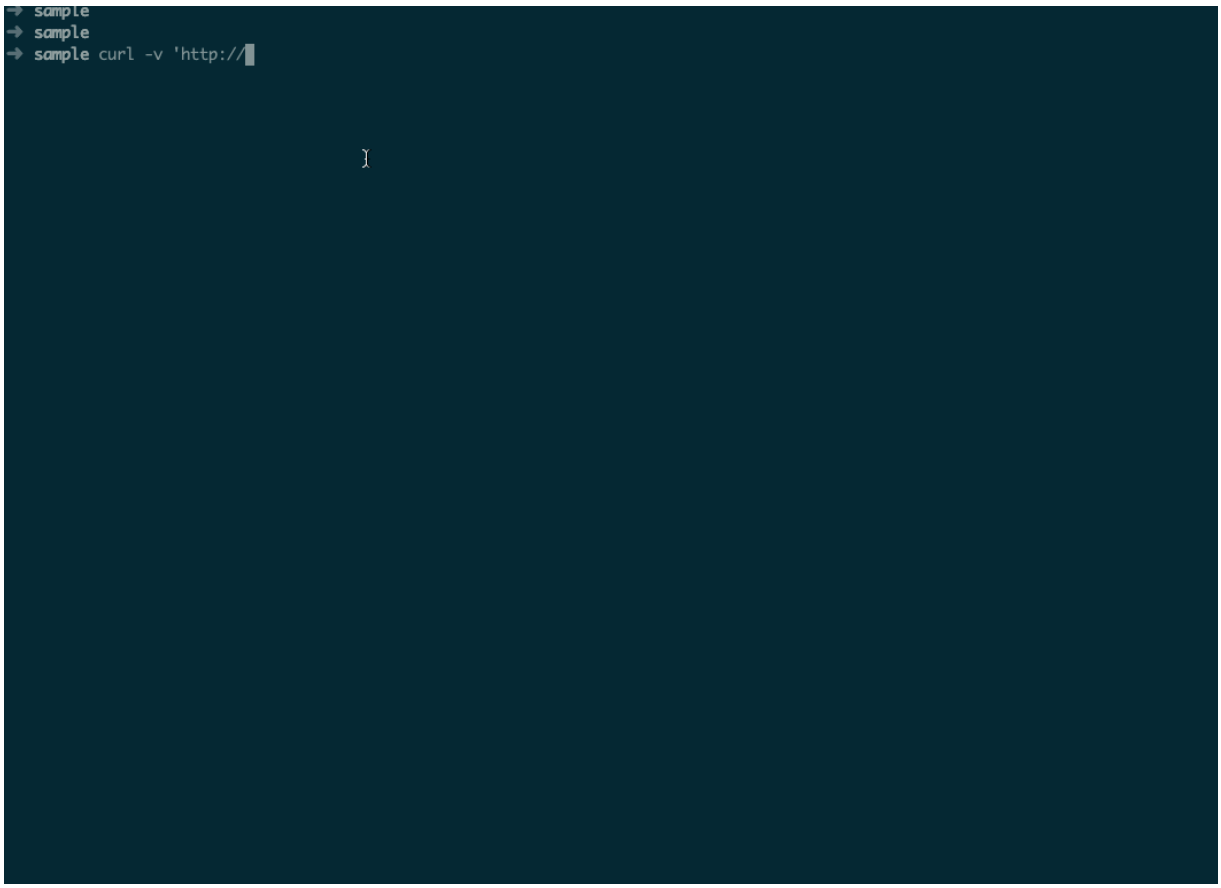


Request

需求：该示例场景实现一个简单的请求添加回源头功能。

命令：

```
curl -v 'http://edge.ialicdn.com/?x=y' -d '{"name":"request", "headers":{"aa":"bb"}, "body":"Hello ER!"}'
```

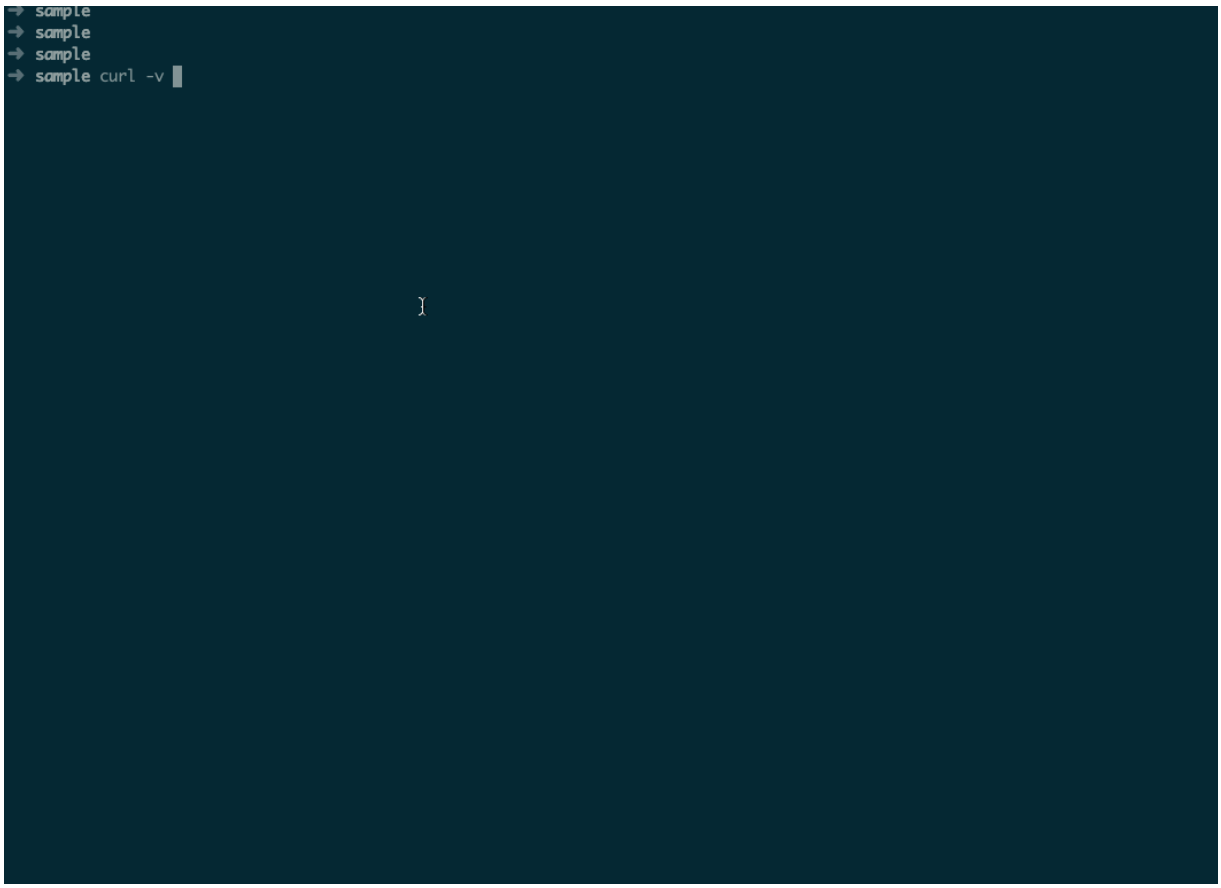


Response

需求：该示例场景实现一个简单的添加回源响应头功能。

命令：

```
curl -v 'http://edge.ialicdn.com/?xx=yy' -d '{"name":"response", "headers":{"ra":"rb"}}'
```

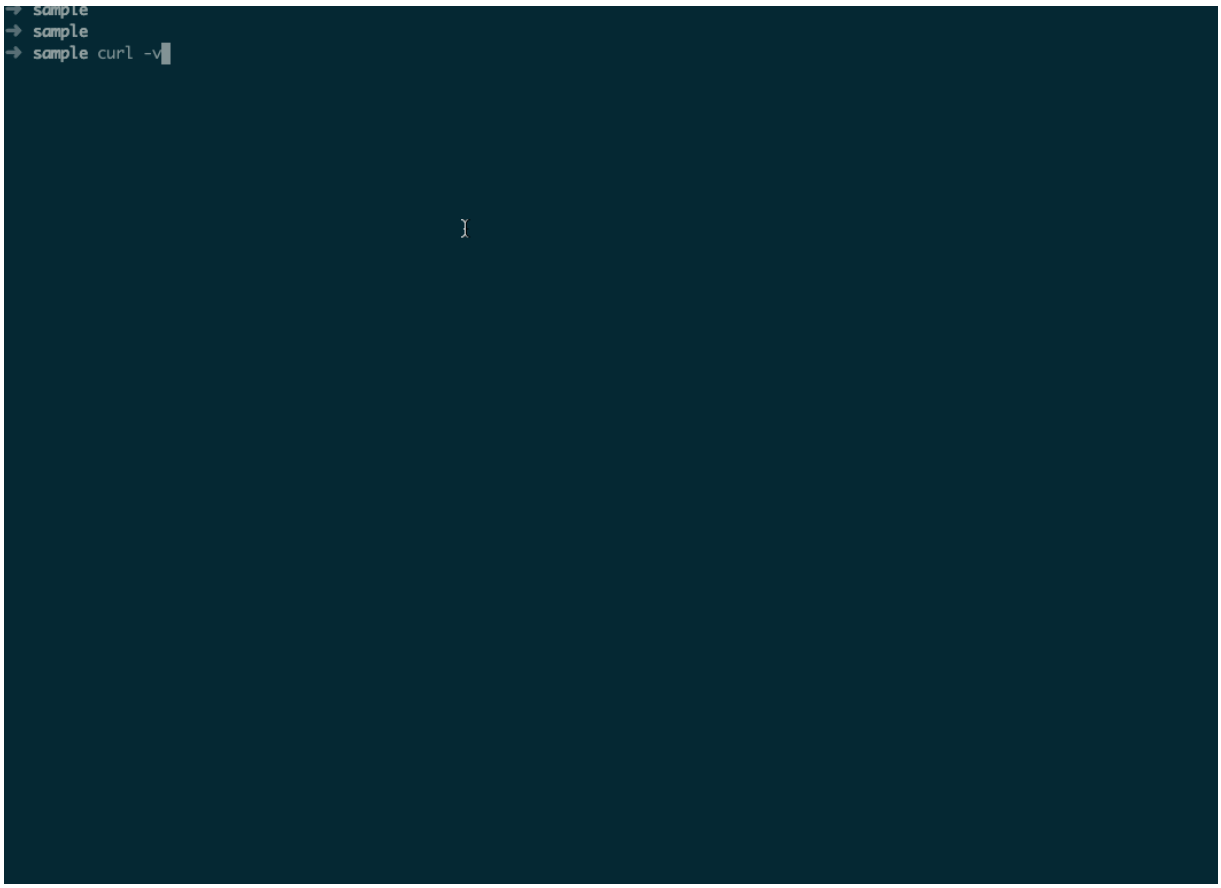


AB test

需求：该示例场景实现一个简单的AB测试的功能。

命令：

```
curl -v 'http://edge.ialicdn.com/?x=y' -d '{"name":"ab-test"}' -H "user-agent: a/canary-client/b"
```

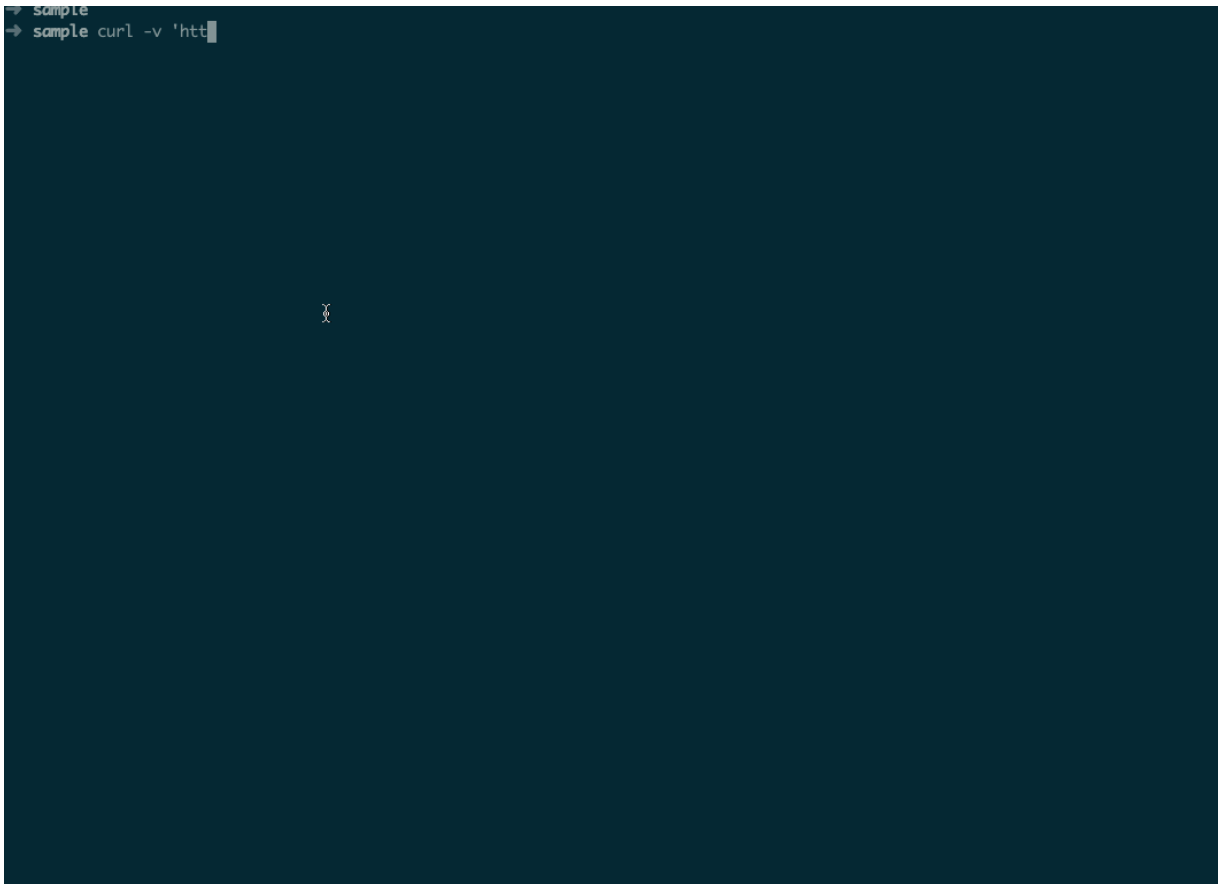


Multi origin

需求：该示例场景实现一个简单的边缘同拉多源合并功能，将不同源站的网页内容聚合后返回给客户端。

命令：

```
curl -v 'http://edge.ialicdn.com/?x=y' -d '{"name":"multi-origin"}'
```

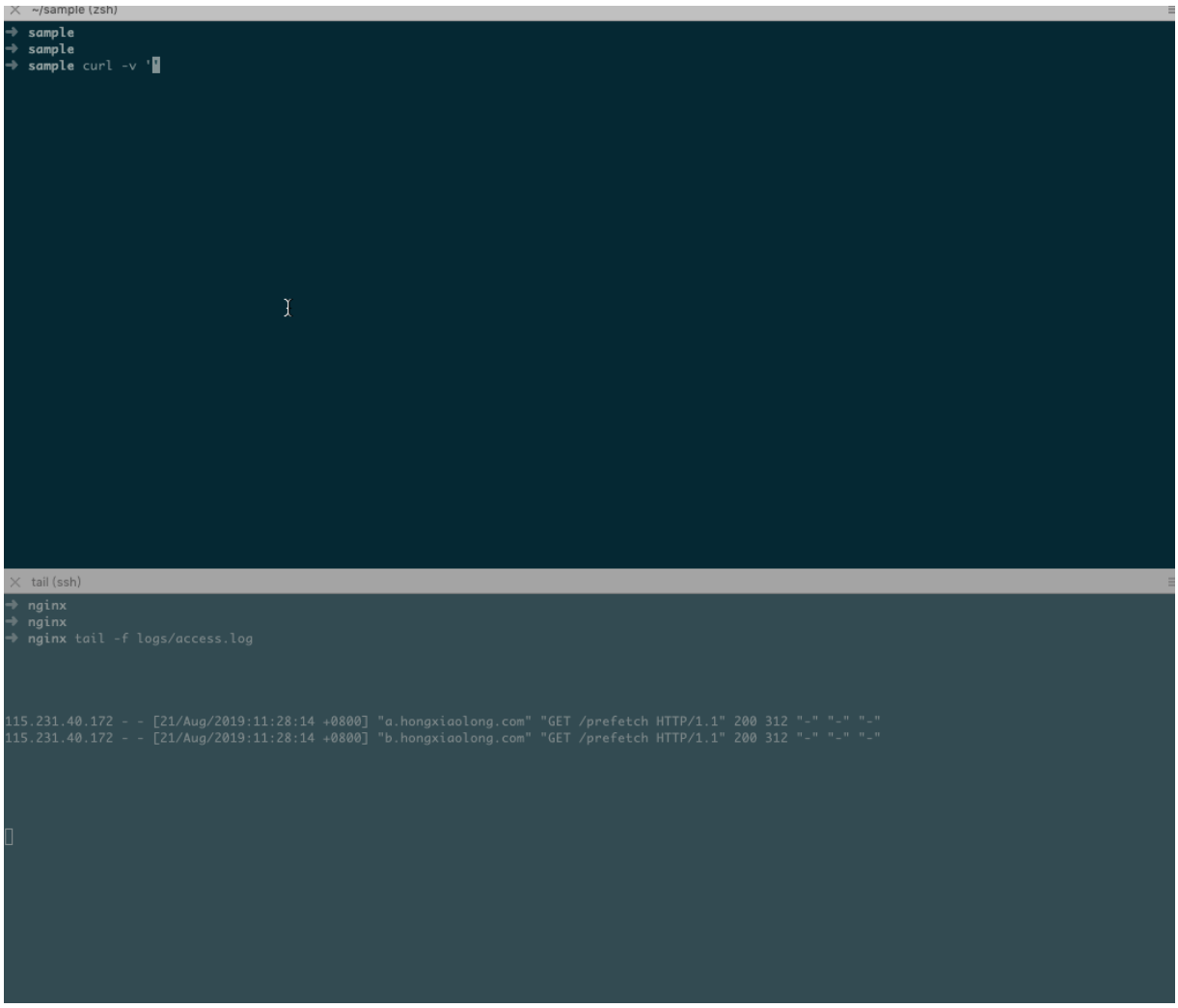


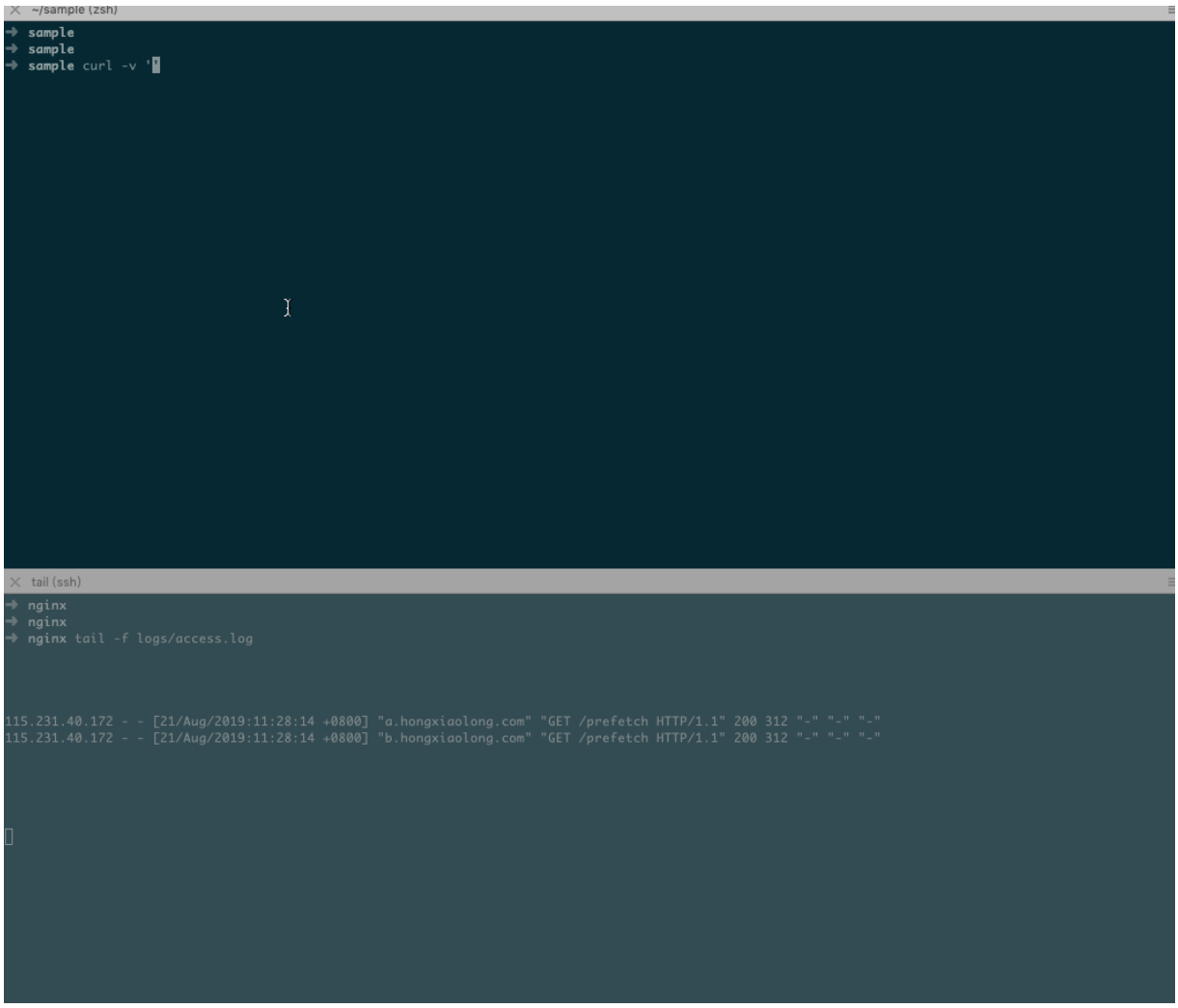
Precache/Prefetch

需求：该示例场景实现一个简单的(CDN)预热功能，预热任务在响应客户端时将异步完成（需下个版本支持，目前忽略)。

命令：

```
curl -v 'http://edge.ialicdn.com/' -d '{"name": "prefetch", "prefetch": ["http://a.hongxiaolong.com/prefetch", "http://b.hongxiaolong.com/prefetch"]}'
```


~/sample (zsh)
sample
sample
sample curl -v '
tail (ssh)
nginx
nginx
nginx tail -f logs/access.log
115.231.40.172 - - [21/Aug/2019:11:28:14 +0800] "a.hongxiaolong.com" "GET /prefetch HTTP/1.1" 200 312 "-" "-" "-"
115.231.40.172 - - [21/Aug/2019:11:28:14 +0800] "b.hongxiaolong.com" "GET /prefetch HTTP/1.1" 200 312 "-" "-" "-"

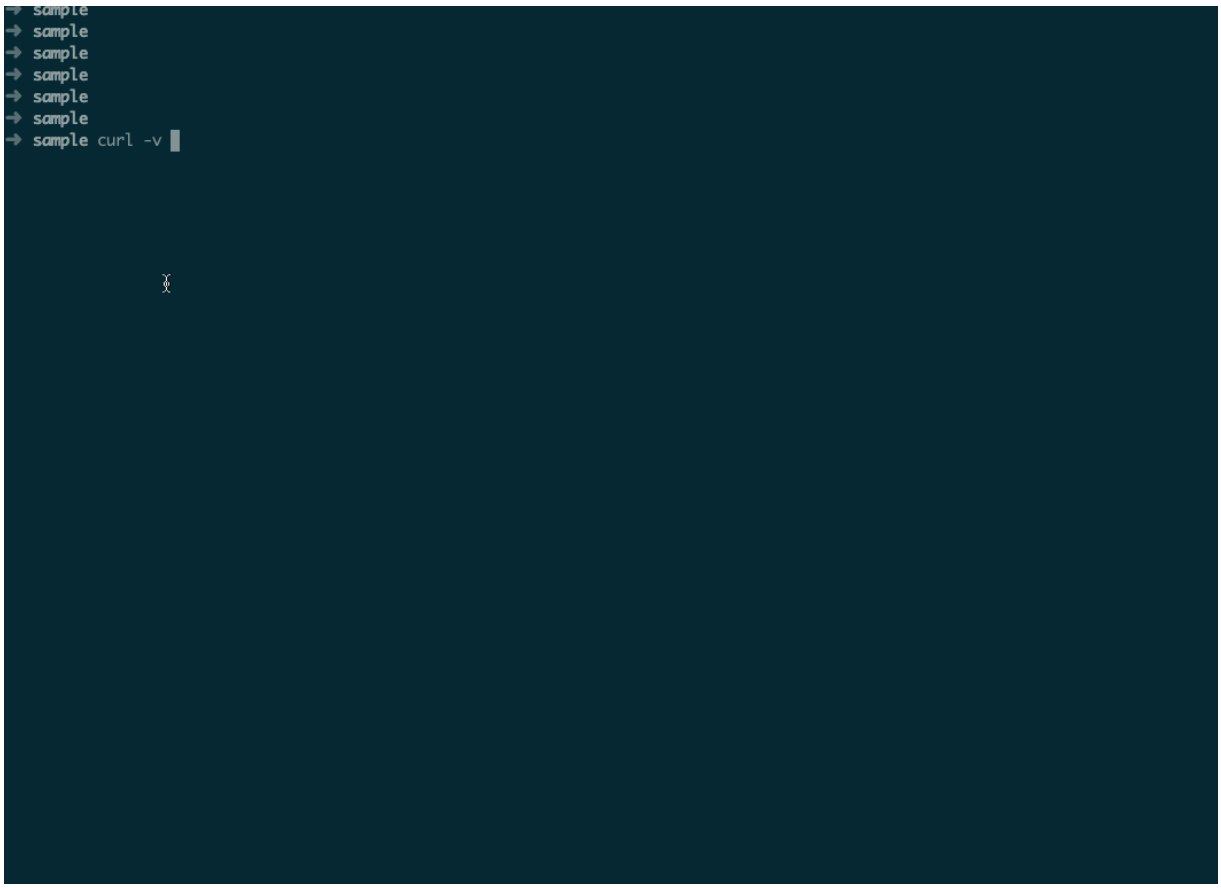


Race

需求：该示例场景实现一个简单的回源同拉功能，将回源速度最快的源站的内容优先返回给客户端。

命令：

```
curl -v 'http://edge.ialicdn.com/?x=y' -d '{"name":"race", "fetchList" : [ "https://www.taobao.com", "https://www.tmall.com", "https://www.baidu.com" ]}'
```

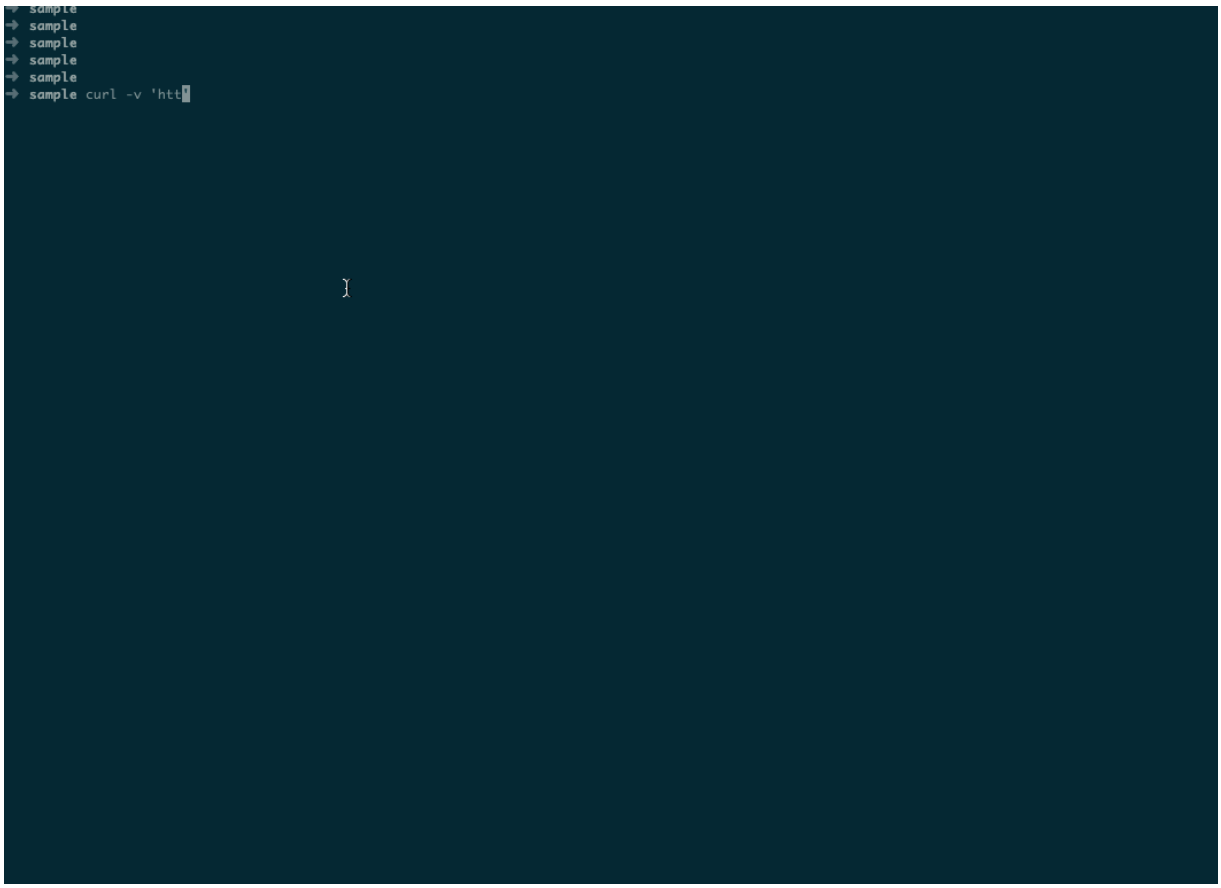


ESI

需求：该示例场景实现一个简单的ESI服务。

命令：

```
curl -v 'http://edge.ialicdn.com/' -d '{"name":"esi","esi" : "<esi:include src=@http://www.baidu.com@>
This is after the ESI for www.baidu.com"}'
```

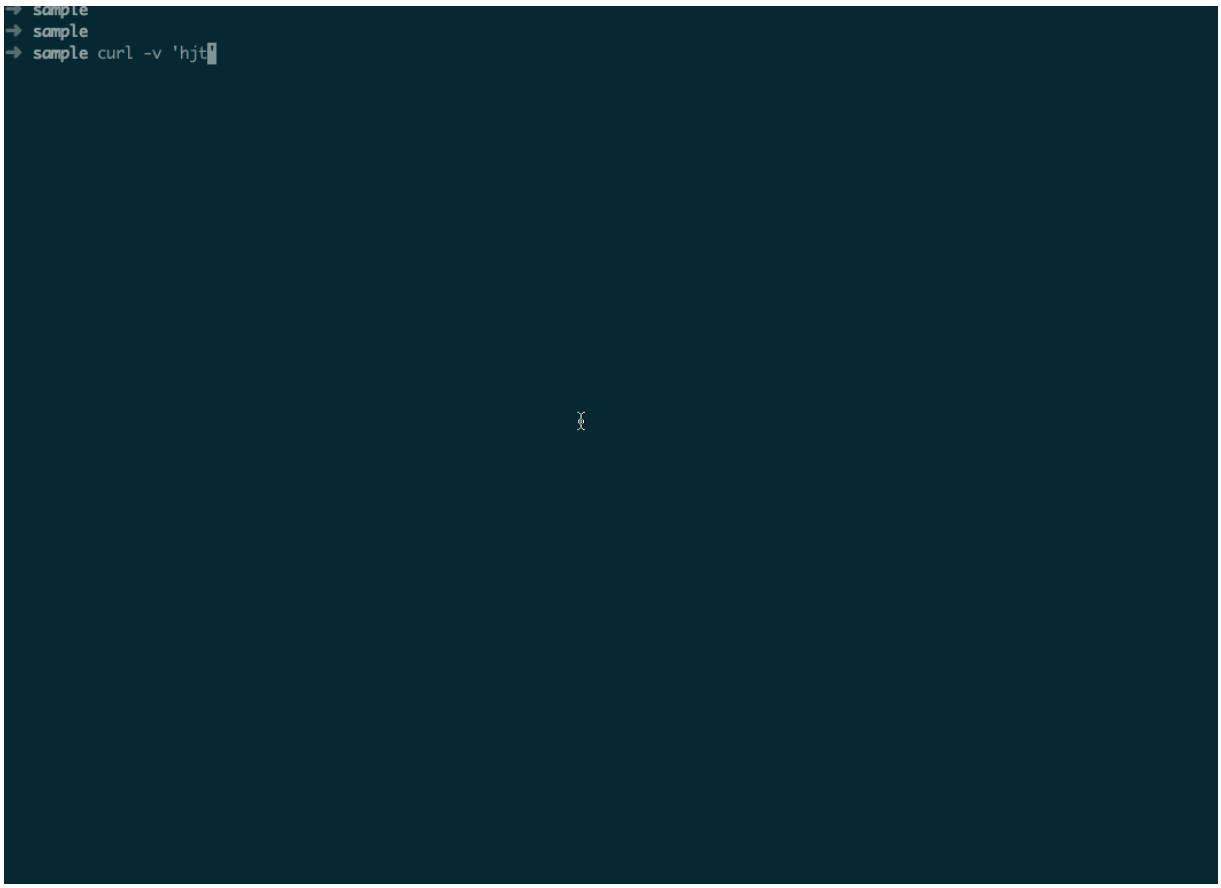


Log

需求：该示例场景实现一个简单的边缘日志服务，在响应结束后异步地生成日志并回传给您的Server。

命令：

```
curl -v 'http://edge.ialicdn.com/' -d '{"name":"log"}'
```

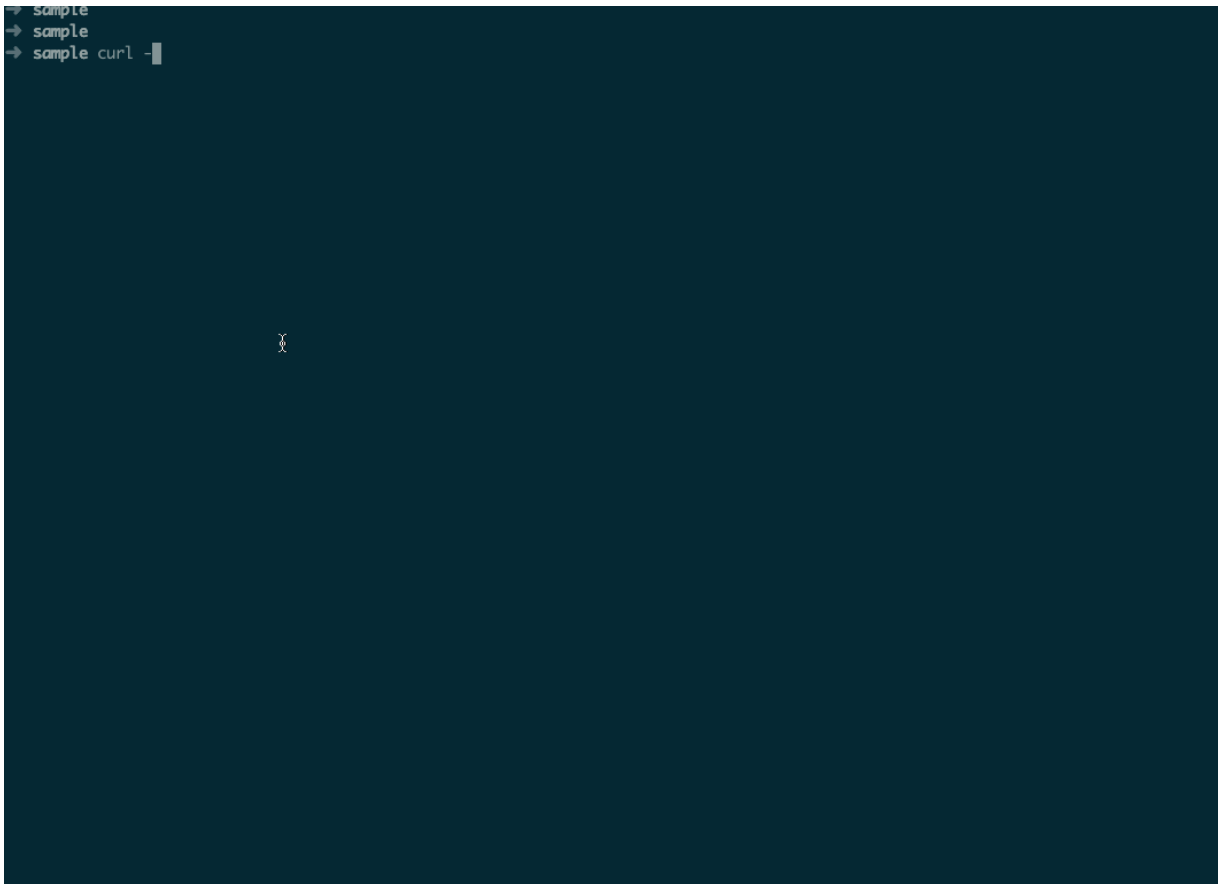


3xx

需求：该示例场景实现一个简单的回源302跟随功能。

命令：

```
curl -v 'http://edge.ialicdn.com/' -d '{"name":"3xx"}'
```

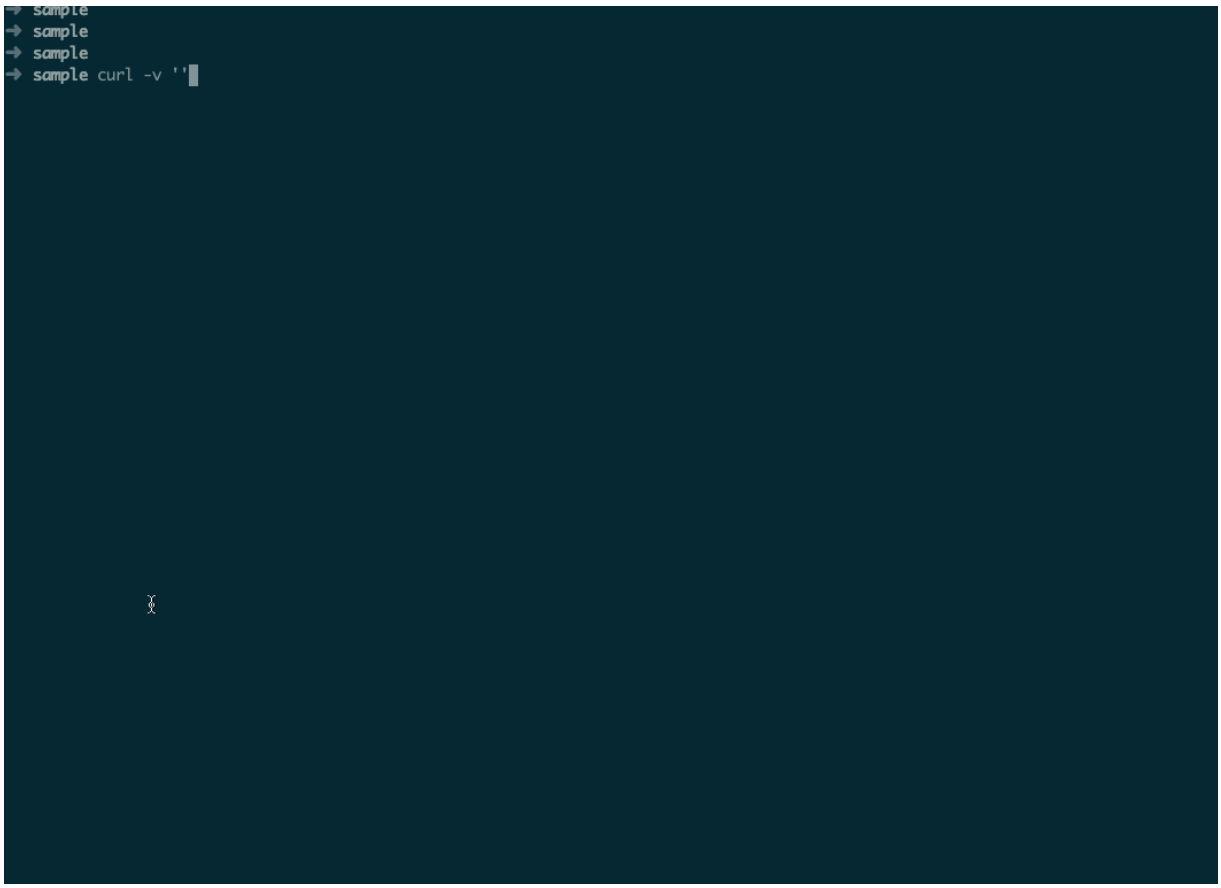


Redirect

需求：该示例场景实现一个简单的边缘请求重定向功能。

命令：

```
curl -v 'http://edge.ialicdn.com/a/b?x=y' -d '{"name": "redirect"}'
```

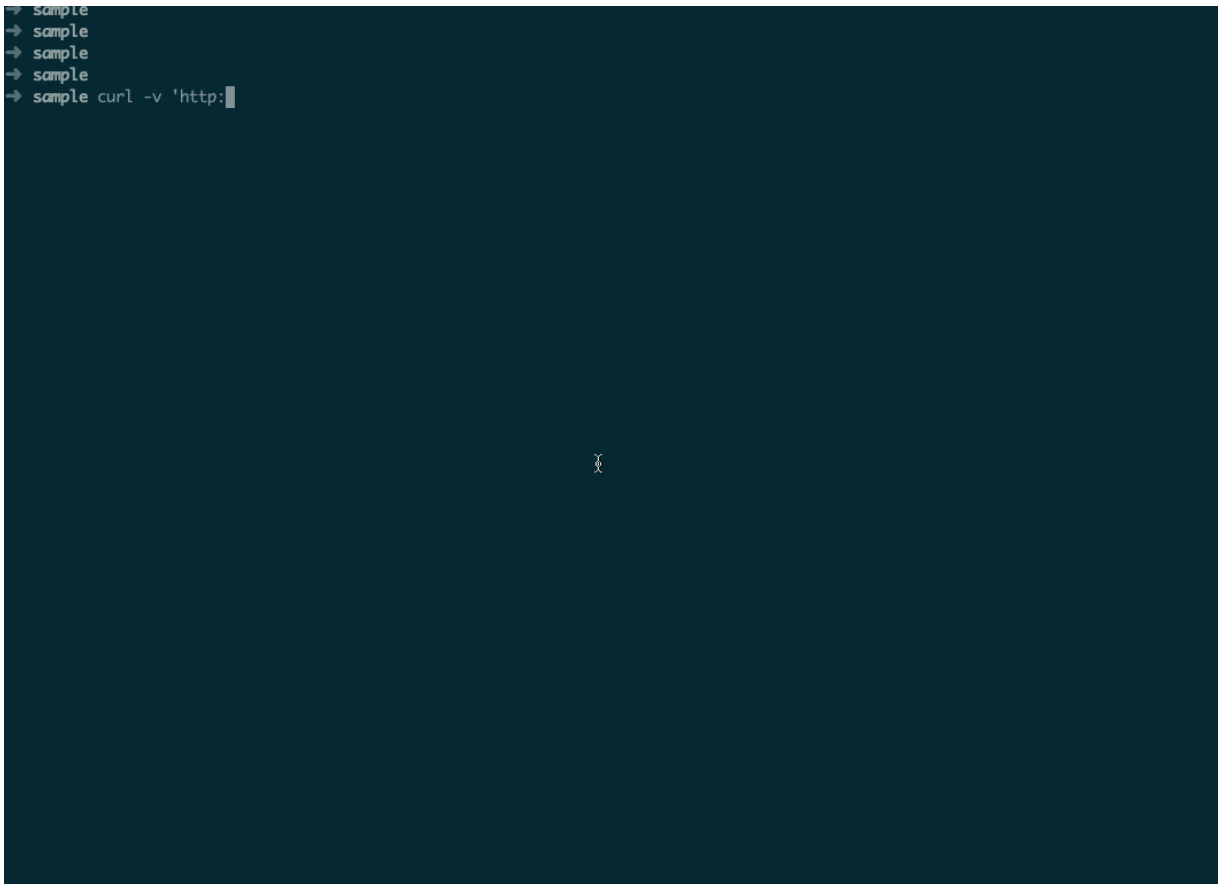


Deny bot

需求：该示例场景实现一个简单的边缘反爬虫服务。

命令：

```
curl -v 'http://edge.ialicdn.com' -d '{"name":"deny-bot"}' -H "user-agent: xxxspider"
```

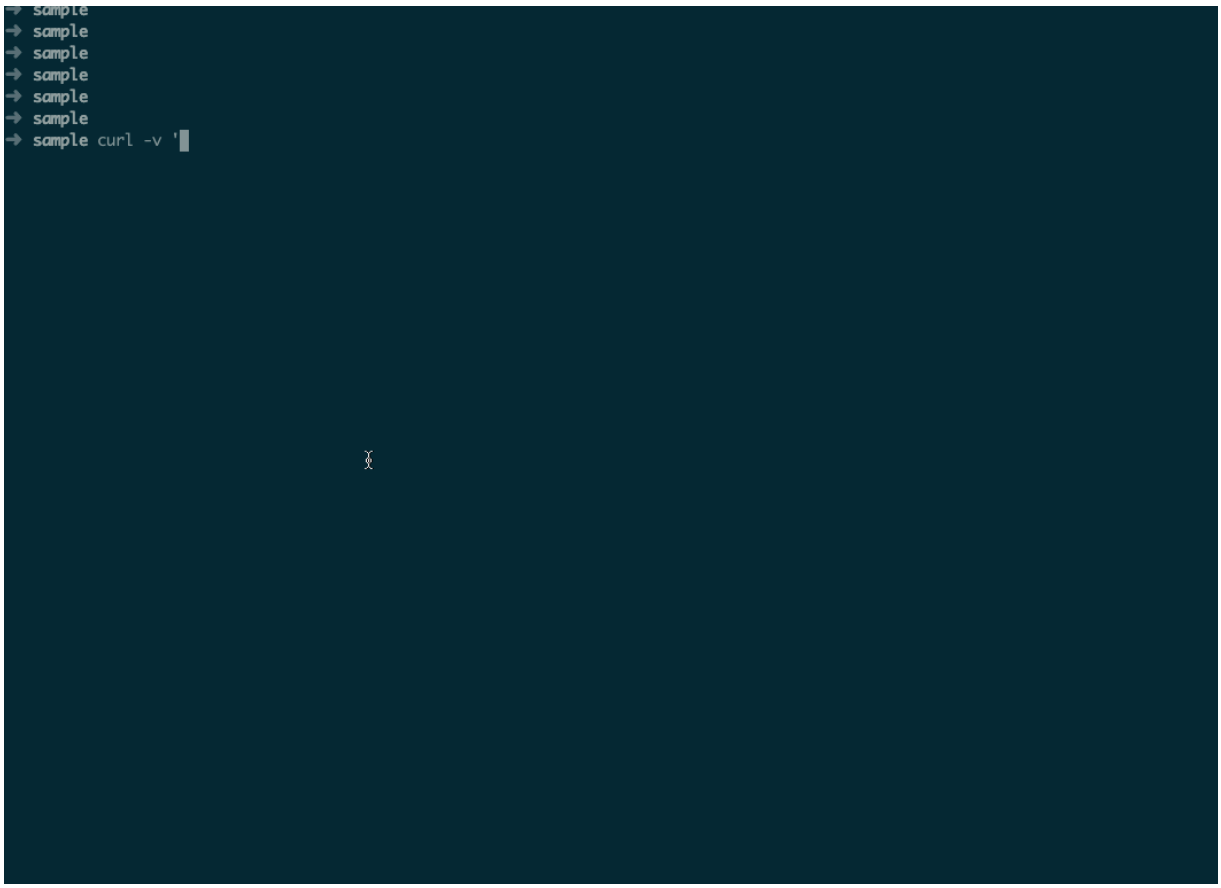


Waf

需求：该示例场景实现一个简单的边缘waf服务，当满足某些条件时，将禁止该请求。

命令：

```
curl -v 'http://edge.ialicdn.com' -d '{"name":"waf", "city":"Hangzhou"}'
```



示例源码

```
addEventListener("fetch", function(event) {
  event.respondWith(_handleRouter(event));
});
async function _handleRouter(event) {
  let json = await event.request.json();
  if (json) {
    let name = json.name
      switch(name) {
        case "helloworld":
          return _handleHelloWorld(event);
        case "geo":
          return _handleGeo(event);
        case "fetch":
          return _handleFetch(event, json);
        case "request":
          return _handleRequest(event, json);
        case "response":
          return _handleResponse(event, json);
        // Cases
```

```
      case "ab-test":
       return _handleABTest(event, json);
      case "multi-origin":
       return _handleMultipleOriginConcate(event, json);
      case "prefetch":
       return _handlePrefetch(event, json);
      case "race":
       return _handleRace(event, json);
      case "esi":
       return _handleESI(event, json);
     case "log":
       return _handleEdgeLog(event, json);
      case "3xx":
       return _handleRedirect3XX(event, json);
      case "redirect":
       return _handleRedirectGeneral(event, json);
      case "deny-bot":
       return _handleDenyBot(event, json);
      case "waf":
       return _handleWAF(event, json);
      default:
       break;
    }
  }
  return new Response(
   `{ error : "invalid request" }`,
   {
    "status" : 403 ,
     "statusText" : "Forbidden"
   });
}
async function _handleHelloWorld(event) {
  return new Response("Hello World!");
}
async function _handleGeo(event) {
  const info = event.info;
  let remote_addr = info.remote_addr;
  let ip_isp_en = info.ip_isp_en;
  let ip_city_en = info.ip_city_en;
  let ip_region_en = info.ip_region_en;
  let ip_country_en = info.ip_country_en;
```

```
  let ip_country_en = info.ip_country_en;
  let scheme = info.scheme;
  let detector_device = info.detector_device;
  let content = `Geo: ${remote_addr}, \
                ${ip_isp_en},   \
                ${ip_country_en},   \
                ${ip_city_en},  \
                ${ip_region_en},\
                ${scheme},      \
                ${detector_device}`;
  return new Response(content);
}
async function _handleFetch(event, json) {
  let fetchURL = json.url;
  if (fetchURL) {
    return await fetch(fetchURL);
  }
  return fetch("http://default.ialicdn.com");
}
async function _handleRequest(event, json) {
  let headers = json.headers;
  let body = json.body;
  const fetchInit = {
    body : body,
    headers: headers
  };
  return fetch("http://default.ialicdn.com", fetchInit);
}
async function _handleResponse(event, json) {
  let resp = await fetch("http://default.ialicdn.com");
  let headers = json.headers;
  for (var k in headers) {
    resp.headers.set(k, headers[k]);
  }
  return resp;
}
/** =================*
 * (1) DevOps      |
 * =================*/
function _shouldDoABTest(request) {
  // (1) if request's user agent match a certain string
```

```
  {
    const ua = request.headers.get("user-agent");
    if (ua && ua.match(/canary-client/)) {
      return true;
    }
  }
  // (2) whether we have special header
  {
    return request.headers.has("x-ab-test");
  }
}
async function _handleABTest(event, json) {
  const fetchInit = {
    method : event.request.method,
    headers: event.request.headers,
    body : "empty"
  };
  if (_shouldDoABTest(event.request)) {
    return fetch("http://default.ialicdn.com/dev", fetchInit);
  } else {
    return fetch("http://default.ialicdn.com", fetchInit);
  }
}
/** ==================================*
 * (2) Multiple Origin Concatenation  |
 ** ==================================*/
async function _handleMultipleOriginConcate(event, json) {
  const respInit = {
    headers: event.request.headers,
    body : json.body
  };
  // (1) We try to concate www.baidu.com and www.tmall.com together
  let {readable, writable} = new TransformStream();
  async function controller() {
    let r1 = await fetch("http://www.baidu.com");
    let r2 = await fetch("https://www.tmall.com");
    await r1.body.pipeTo(writable, {preventClose: true});
    await r2.body.pipeTo(writable);
  }
  controller();
  return new Response(readable, respInit);
```

```
}
/** ====================================*
 * (3) Precache/Prefetch            |
 ** ====================================*/
async function _fetchAndIgnore(url) {
 try {
  // Specify cdnProxy flag to make sure the request goes through the CDN
  let resp = await fetch(url);//, {cdnProxy: true});
  // Make sure to ignore the content otherwise the cache may not be valid
  await resp.ignore();
 } catch (e) {
  console.error("invalid URL: %s", url);
 }
}
async function _doPrefetchURLAsync(prefetchURL, event) {
 for (const url of prefetchURL) {
  event.waitUntil(_fetchAndIgnore(url));
 }
}
async function _handlePrefetch(event, json) {
 {
  const prefetchURL = json.prefetch;
  if (prefetchURL) {
   // Do not await it and let it run in background
   _doPrefetchURLAsync(prefetchURL, event);
   return new Response("Done Prefetch");
  }
 }
 return new Response("Miss Prefetch");
}
/** ====================================*
 * (4) Race
 ** ====================================*/
async function _handleRace(event, json) {
 let fetchList = json.fetchList;
 if (fetchList) {
  return Promise.race(fetchList.map((x) => fetch(x)));
 } else {
  return "forget to include fetchList field in your JSON";
 }
}
```

```
}
/** ==================================*
 * (5) Simple ESI
 ** ==================================*/
async function _handleESI(request, json) {
  let { readable, writable } = new TransformStream();
  let newResponse = new Response(readable);
  if (!json.esi) {
    return "forget to include template field in your JSON";
  }
  streamTransformBody(new BufferStream(json.esi), writable);
  return newResponse;
}
async function handleTemplate(encoder, templateKey) {
  const linkRegex = new RegExp("esi:include.*src=@(.*)@.*", 'gm');
  let result = linkRegex.exec(templateKey);
  let esi = "unknown";
  if (!result) {
    return encoder.encode(`<${templateKey}>`);
  }
  if (result[1]) {
    esi = await subRequests(result[1]);
  }
  return encoder.encode(`${esi}`);
}
async function subRequests(target){
  const init = {method: 'GET'};
  let response = await fetch(target, init);
  let text = await response.text();
  return text;
}
async function streamTransformBody(readable, writable) {
  const startTag = "<".charCodeAt(0);
  const endTag = ">".charCodeAt(0);
  let reader = readable.getReader();
  let writer = writable.getWriter();
  let templateChunks = null;
  while (true) {
    let { done, value } = await reader.read();
    if (done) break;
    while (value.byteLength > 0) {
```

```
    if (templateChunks) {
      let end = value.indexOf(endTag);
      if (end === -1) {
        templateChunks.push(value);
        break;
      } else {
        templateChunks.push(value.subarray(0, end));
        await writer.write(await translate(templateChunks));
        templateChunks = null;
        value = value.subarray(end + 1);
      }
    }
    let start = value.indexOf(startTag);
    if (start === -1) {
      await writer.write(value);
      break;
    } else {
      await writer.write(value.subarray(0, start));
      value = value.subarray(start + 1);
      templateChunks = [];
    }
   }
  }
  await writer.close();
}
async function translate(chunks) {
  const decoder = new TextDecoder();
  let templateKey = chunks.reduce(
    (accumulator, chunk) =>
    accumulator + decoder.decode(chunk, { stream: true }), "");
  templateKey += decoder.decode();
  return handleTemplate(new TextEncoder(), templateKey);
}
/** ==================================*
 * (6) Edge side conditional log
 ** ==================================*/
async function _doEdgeLog(data, writer) {
  let resp = await fetch("http://default.ialicdn.com/log",
    {
      method : "POST",
      body   : data,
```

```
    headers: [["content-type", "application/json"]]
   });
  console.log("logged");
  {
   let stream = new BufferStream("++++++++++++++++++++++++++++++\n");
   await stream.pipeTo(writer, {preventClose: true});
  }
  await resp.body.pipeTo(writer);
}
async function _handleEdgeLog(event, json) {
  let start= Date.now();
  let resp = await fetch("http://default.ialicdn.com", {
   method : event.request.method,
   headers: event.request.headers,
   body : json.body
  });
  // Get a promise that is fired when we send out everything
  let {readable, writable} = new TransformStream();
  // (1) first let the fetch request's response goes back and then we post
  //    the log back as well internally
  let endPromise = resp.body.pipeTo(writable, {preventClose: true});
  // (2) wait for endPromise to be fired to make sure that the body has been
  //    piped back to the client, and then we do the log
  event.waitUntil(endPromise.then(
   (v) => {
    let end = Date.now();
    let diff= (end - start);
    try {
     // You have to await your async promise since wait until is not
     // usable currently maybe. User can use wait until only before
     // returning the main request for now
     event.waitUntil(_doEdgeLog(`{ "cost(millisecond)" : ${diff} }`, writable));
    } catch (e) {
     console.error(`${e}`);
    }
   },
   (v) => {
    writable.abort();
    console.error("failed");
   }));
```

```
  console.error("XXXX");
  // return the response back
  return new Response(readable, {
    status: resp.status,
    headers: resp.headers
  });
}
/** ==================================*
 * (7) redirect-3xx
 ** ==================================*/
async function _handleRedirect3XX(event, json) {
  return fetch("http://www.taobao.com", {redirect: "follow"});
}
/** ==================================*
 * (8) redirect
 *   (1) UserAgent
 *   (2) Geo information
 ** ==================================*/
async function _handleRedirectGeneral(event, json) {
  const fetchInit = {
    method : event.request.method,
    body : json.body,
    headers : event.request.headers
  };
  {
    const ua = event.request.headers.get("user-agent");
    if (ua && ua.match(/firefox/i)) {
      return fetch("http://default.ialicdn.com/firefox", fetchInit);
    }
    if (ua && ua.match(/safari/i)) {
      return fetch("http://default.ialicdn.com/safari", fetchInit);
    }
  }
  {
    if (event.info.detect_device && event.info.detect_device.match(/iphone/)) {
      return fetch("http://default.ialicdn.com/iphone", fetchInit);
    }
  }
  return new Response("unknown request", {status: 403});
}
/** ==================================*
```

```
 * (9) Deny bot
 ** ===================================*/
async function _handleDenyBot(event, json) {
 {
   const ua = event.request.headers.get("user-agent");
   if (ua && ua.match(new RegExp("xxxspider", "i"))) {
     return new Response("Forbidden", {status: 403});
   }
 }
 return fetch("http://default.ialicdn.com");
}
/** ====================================*
 * (10) Simple WAF
 ** ===================================*/
async function _handleWAF(event, json) {
 let city = json.city;
 if (event.info.ip_city_en === city) {
   return new Response("Forbidden", {status: 403});
 }
 // back to origin
 return (JSON.stringify(event.info));
}
```

3.EdgeRoutine CLI工具使用说明

阿里云CDN为您提供EdgeRoutine CLI工具，可以直接在CDN节点执行您自行编写的JavaScript代码，通过本章节，您可以了解使用EdgeRoutine CLI工具发布代码的具体步骤。

前提条件

EdgeRoutine CLI工具运行环境需要Node.js 8.0及以上版本。

EdgeRoutine需要将边缘Serverless环境绑定至您的CDN加速域名，依赖您的阿里云账号来进行相关配置，所以您需要在控制台完成添加加速域名，具体操作请参见添加加速域名。获取AccessKeyID和AccessKeySecret，请参见。

目前EdgeRoutine处于邀测阶段，您需要提交工单开通。

操作步骤

1. 安装EdgeRoutine CLI工具。 在终端中运行以下命令进行安装。

   ```
   $ npm install @alicloud/edgeroutine-cli -g
   ```

2. 创建代码目录并切换到已创建目录。

   ```
   $ mkdir yourProject; cd yourProject
   ```

3. 初始化环境并创建模板代码。 初始化环境命令如下所示。

   ```
   $ edgeroutine-cli init
   ```

   代码模板示例如下，如果您已有目标代码，您可以将您的代码文件移动到当前目录命名为*edge.js*即可。

   ```
   /**
    * Add the necessary event listener
    * @param {Event} fetch event, {Function} async function
    */
   addEventListener('fetch', event => {
    event.respondWith(handleRequest(event.request))
   })
   /**
    * Make a response to client
    * @param {Request} request
    */
   async function handleRequest(request) {
    return new Response('Hello World!', { status: 200 })
   }
   ```

4. 配置域名和AK信息（AccessKeyID和AccessKeySecret）。

   ```
   $ edgeroutine-cli config
   ```

5. 代码发布到灰度环境。

   > 说明 在您将代码发布到生产环境之前必须执行验证灰度环境。

   i. 创建灰度环境代码。

   ```
   $ edgeroutine-cli build
   ```

   ii. 查看灰度环境代码。

   ```
   $ edgeroutine-cli build --show
   ```

   iii. 删除灰度环境代码。

   ```
   $ edgeroutine-cli build --delete
   ```

   iv. 回滚灰度环境代码。

   您可以随时继续发布，但此时您无法将空代码发布到生产环境，请注意回滚和删除的区别。

   ```
   $ edgeroutine-cli build --rollback
   ```

6. 测试灰度环境代码。

   请您在终端环境中访问目标域名，EdgeRoutine灰度环境节点IP：42.123.119.50或42.123.119.51。

   ```
   $ curl-v'http://yourdomain.com/yourpath/'-x42.123.119.50:80
   ```

7. 发布代码到生产环境。

   i. 当您完成验证灰度环境时，您可将灰度环境的代码发布到线上环境。

   ```
   $ edgeroutine-cli publish
   ```

   ii. 查看线上环境代码。

   ```
   $ edgeroutine-cli publish --show
   ```

   iii. （可选）删除线上环境代码，此时将直接彻底删除代码和配置。

   ```
   $ edgeroutine-cli publish --delete
   ```

8. 测试生产环境。 此时您可直接测试目标域名的线上业务。

   ```
   $ curl -v 'https://yourdomain.com/yourpath/'
   ```