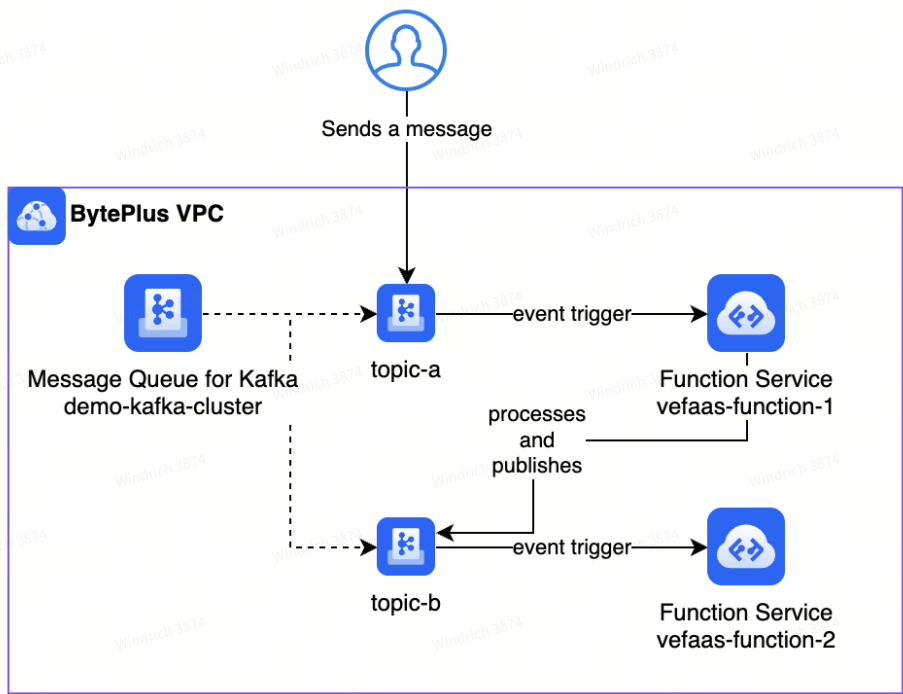


[External] BytePlus Function Service & Kafka Sample Deployment Guide

This deployment creates the necessary resources for the following flow:



Create Message Queue for Kafka cluster & retrieve connection details

Refer to this guide for more information on how to create the Kafka cluster:

<https://docs.byteplus.com/en/docs/kafka/creating-instance>. In this guide, we are using Kafka version 3.7.1, single-AZ deployment.

Once created, retrieve the default access point URL, as well as the SASL_PLAINTEXT access point.

Service access

Public Network Access

1 public IPs bound | [Turn off public network access](#)

Default access point

Private network kafka-ap-1115j3vgwf5vt.kafka.ibebyteplus.com:9092

SASL_SSL access point

Private network kafka-ap-1115j3vgwf5vt.kafka.ibebyteplus.com:9095
Public Network kafka-ap-1115j3vgwf5vt.kafka.ibebyteplus.com:9491

SASL_PLAINTEXT access point

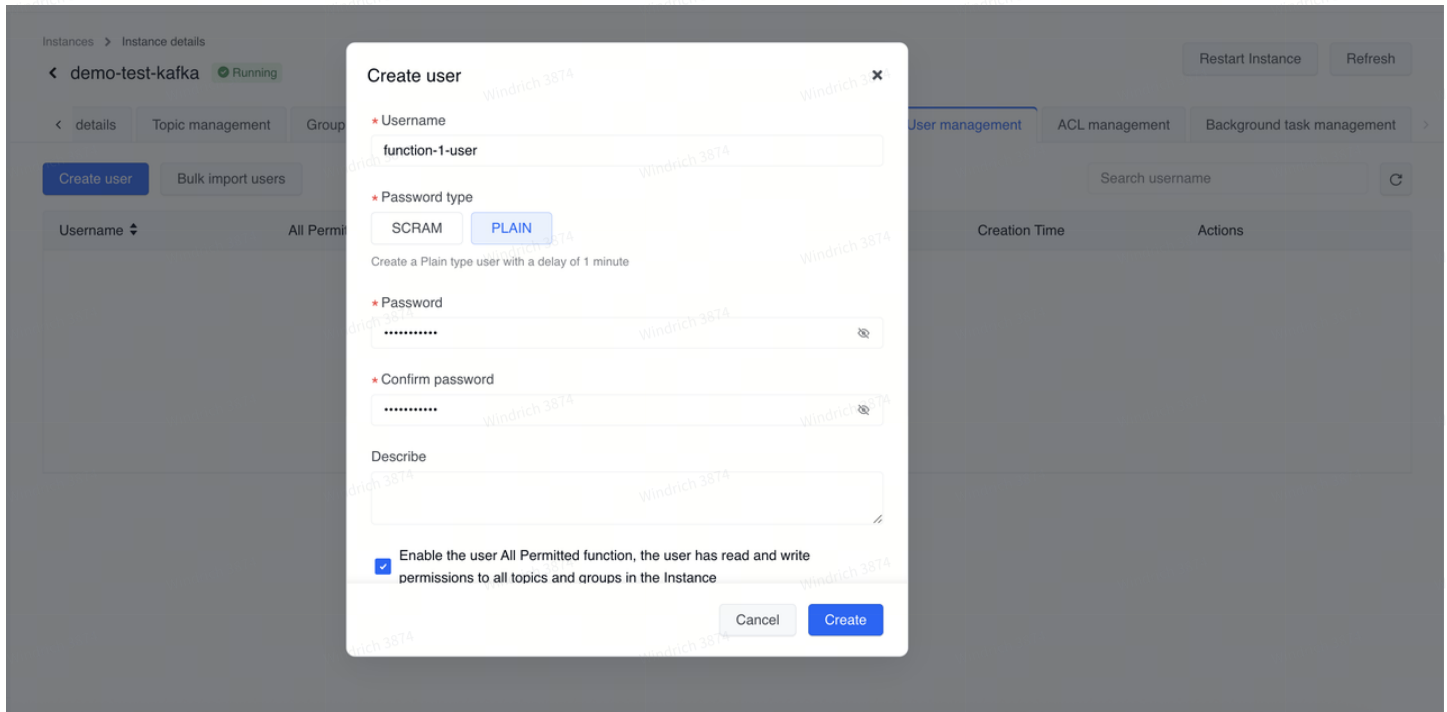
Private network kafka-ap-1115j3vgwf5vt.kafka.ibebyteplus.com:9093
Public Network kafka-ap-1115j3vgwf5vt.kafka.ibebyteplus.com:9591

Create topics in the cluster

Under "Topic management", create 2 topics - " `topic-a` " and " `topic-b` ". Feel free to choose your own topic names.

Create SASL users in Kafka

In the Kafka cluster, go to "User management" tab, and create 2 users - `function-1-user` and `function-2-user` . These credentials will be used by Serverless Functions.



Verify Kafka configuration

Create an ECS instance (refer to this guide for more information:

<https://docs.byteplus.com/en/docs/ecs/Buying-and-using-ECS-instances>) as a client to test connectivity and verify configuration. This ECS instance will also be used to send initial messages to Kafka topic later.

Install Java and Kafka client tool:

Code block

```
1  # Install Java
2  sudo apt update && sudo apt install -y openjdk-17-jre-headless
3
4  # Download and extract Kafka 3.7.2
5  wget https://downloads.apache.org/kafka/3.7.2/kafka_2.13-3.7.2.tgz
6  tar -xzf kafka_2.13-3.7.2.tgz
7  cd kafka_2.13-3.7.2
```

Start consuming messages from topic-a:

```
1 bin/kafka-console-consumer.sh --bootstrap-server <BROKER_ENDPOINT>:9092 --  
topic topic-a --from-beginning
```

Open a new terminal connected to the same instance, and send a test message to topic-a:

Code block

```
1 bin/kafka-console-producer.sh \  
2 --broker-list <BROKER_ENDPOINT>:9092 \  
3 --topic topic-a  
4 >{"user":"alice","action":"login"}
```

The messages should appear on the consumer side.

Create Serverless Functions

Create vefaas-function-1

Go to BytePlus Function Service console and create a new function. Choose "Event function" as the type, input the name `vefaas-function-1` and choose `Node.js 20.x` for the runtime. Keep the deployment method to "Function template".

< Create function

Select the method to create a function

Create 「Event function」
Event-driven function computing with p...

Create 「Web applications」
Web applications deployed via contain...

Create 「Microservices applications」
Microservices applications deployed vi...

Create 「Task」
Respond to invocation requests in task...

Basic information

Name *
vefaas-function-1
The name cannot be modified after creation

Description
Please enter
0/200

Function code

Runtime *
Node.js 20.x

Deployment method *
Function template Upload local file Upload TOS

Under Network, enable it and choose the VPC and subnets that are in the same VPC as the Kafka cluster. This enables the functions to communicate with the Kafka cluster privately without going through the Internet. Also, enable public network access so that it can reach the Internet to download and install packages.

< Create function

Network ⓘ

☒ Enable

After VPC is enabled and configured, functions run and access resources within the specified VPC.

Virtual private cloud *

win-vpc-1 (10.0.0.0/16)

[Create Virtual Private Cloud](#) | [C](#)

Subnet *

Asia Pacific (Johor) - Availability zone 1a | win-vpc-1-private-subnet...

Asia Pacific (Johor) - Availability zone 1b | win-vpc-1-private-subnet...

[Create subnet](#) | [C](#)

Each Availability Zone can have up to one subnet.

Security group *

Default

Public network access ⓘ

☒ Enable

When enabled, functions can access public network via default network card regardless of VPC.

Leave the rest to default and create the function.

Inside the function, under "Code", update the package.json to the following:

Code block

```
1  {
2    "name": "vefaas-function-1",
3    "version": "1.0.0",
4    "description": "Read and process message from a Kafka topic, and publish to
another Kafka topic",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "dependencies": {
11     "kafkajs": "^2.2.0"
12   }
13 }
```

Update the `index.js` with the following code:

Code block

```
1  const { Kafka } = require('kafkajs');
2
3  const kafka = new Kafka({
4    clientId: 'function1-producer',
5    brokers: [process.env.KAFKA_BOOTSTRAP_SERVER],
6    ssl: false, // Set to true if your Kafka cluster requires SSL
```

```

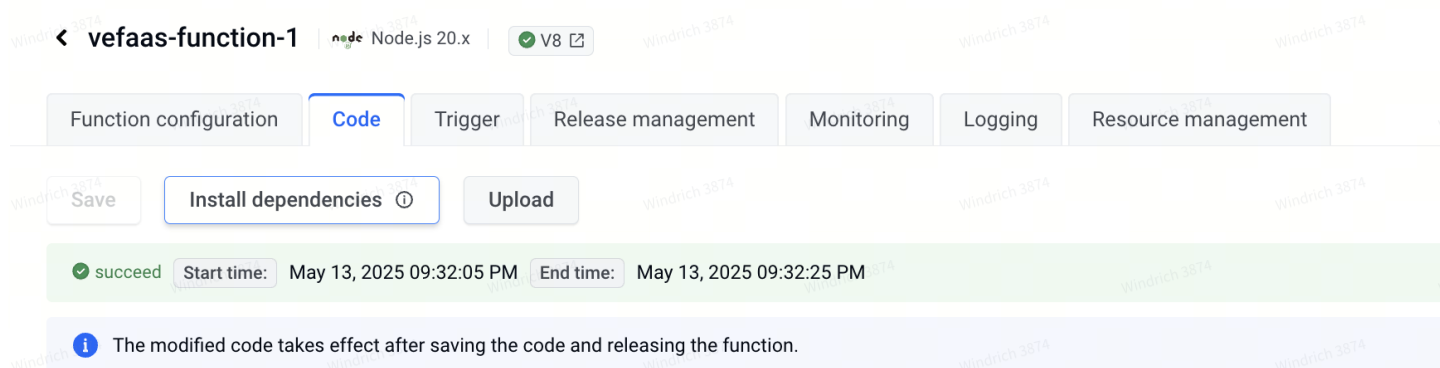
7   sasl: {
8     mechanism: 'plain',
9     username: process.env.KAFKA_USERNAME, // Your Kafka username
10    password: process.env.KAFKA_PASSWORD // Store password in environment
    variable
11  }
12  });
13
14  exports.handler = async function handler(event, context) {
15    const topicName = 'topic-b';
16
17    console.log(`received new request, request id: %s`, context.requestId);
18    console.log(`event: `, event);
19    const message = event.data;
20
21    // Simple processing: append a processedAt timestamp
22    message.processedAt = new Date().toISOString();
23
24    const producer = kafka.producer();
25    await producer.connect();
26    await producer.send({
27      topic: topicName,
28      messages: [{ value: JSON.stringify(message) }],
29    });
30    await producer.disconnect();
31
32    console.log(`Successfully processed and sent this message to ${topicName}:
    "${JSON.stringify(message)}"`);
33
34    return {
35      statusCode: 200,
36      headers: { 'Content-Type': 'application/json' },
37      body: JSON.stringify({ 'message': `message received and processed:
    ${JSON.stringify(message)}` }),
38    };
39  };
40

```

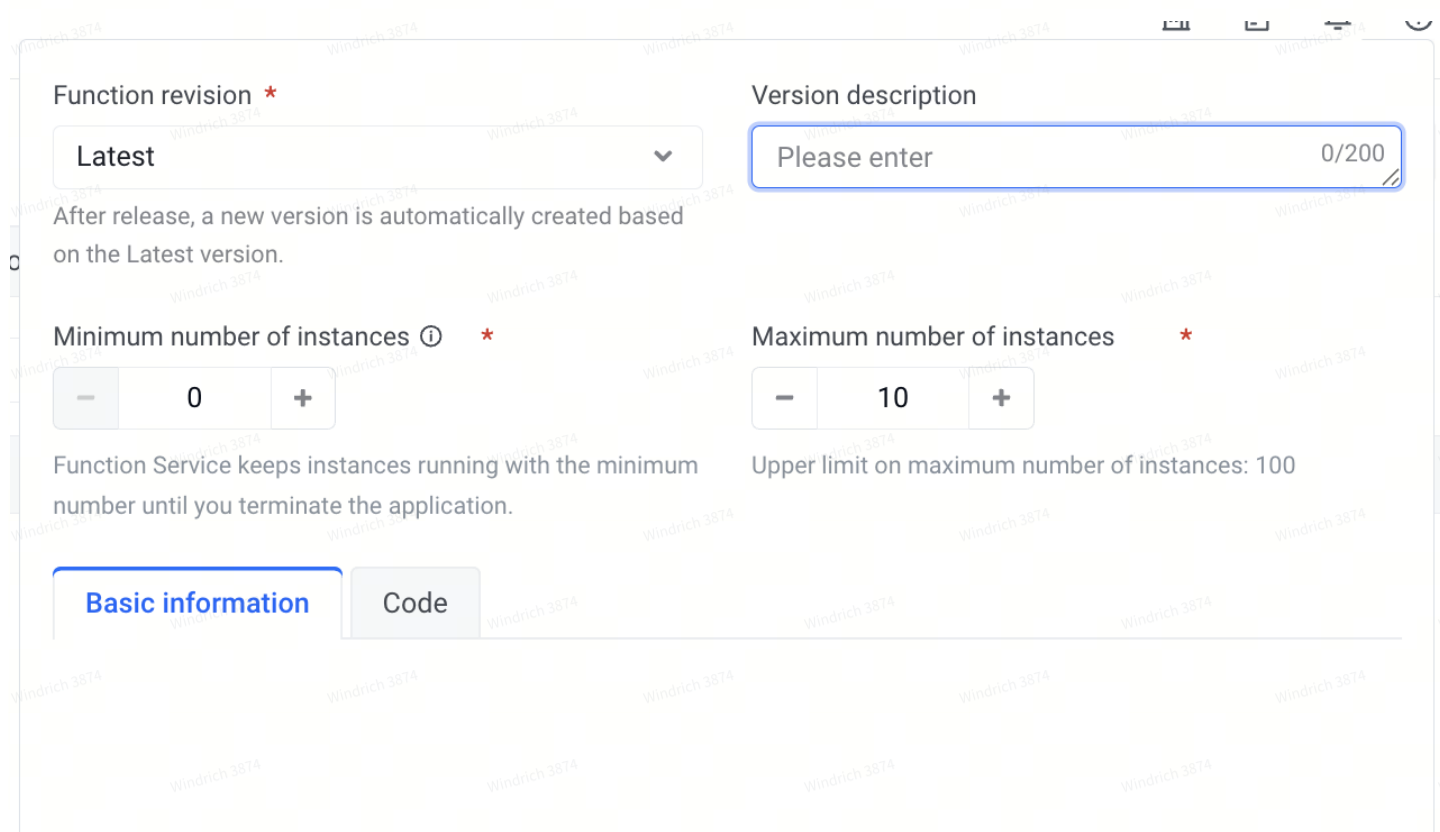
Go to function configuration and under Environment variables, add the variables:

- **KAFKA_BOOTSTRAP_SERVER** with the Kafka cluster SASL_PLAINTEXT URL, e.g. kafka-ap-1115j3vgwf5vt.kafka.iobytepluses.com:9093.
- **KAFKA_USERNAME** with the username for this function, **function-1-user**.
- **KAFKA_PASSWORD** with the password set for the user.

Under Code tab, install the required dependencies by clicking the **Install dependencies** button.



Click the Release button to publish the function. You can keep the parameters to default.



Once the function has been published, go to Trigger tab and create a Kafka trigger (documentation: https://docs.byteplus.com/en/docs/faas/Creating_a_Kafka_trigger).

Create triggerTrigger overviewX

Trigger type *

Message Queue for Kafka

Trigger name *

kafka-trigger-topic-a

The name cannot be modified after the trigger has been successfully created.

Instance *

demo-test-kafka

Create instance

Topic *

topic-a

Create topic

Number of retries *

100

time

Range of retries: 0-100

Consumer offset *

From the beginning offset

From the latest offset

Enable immediately

Create triggerTrigger overviewX

Consumer offset *

From the beginning offset

From the latest offset

Enable immediately

Authentication ⓘ *

Password type

PLAIN

SCRAM-SHA-256

Username

function-1-user

Password

Function Service does not store the username and password, which are only used for authentication after the trigger has been created.

Description

Please enter

CancelCreate

Create vefaas-function-2

Similar to the above steps, create another function named `vefaas-function-2`.

As this function simply prints out messages in `topic-b`, there's no need for additional dependencies or environment variables.

Add the following code for `index.js`, and then Release the function.

Code block

```
1  exports.handler = async function handler(event, context) {
2    console.log(`received new request, request id: %s`, context.requestId);
3    console.log(`Received message from topic-b: `, event.data);
4
5    return {
6      statusCode: 200,
7      headers: { 'Content-Type': 'application/json' },
8      body: JSON.stringify({ 'message': `received message from topic-b:
9      ${event.data}` }},
10   };
};
```

Create the Kafka trigger similar to above, but choose `topic-b` and use the `function-2-user` credentials.

Create trigger

Trigger overview

Trigger type *

Message Queue for Kafka

Trigger name *

kafka-trigger-topic-b

The name cannot be modified after the trigger has been successfully created.

Instance *

demo-test-kafka

Create instance

Topic *

topic-b

Create topic

Number of retries *

5

time

Range of retries: 0-100

Consumer offset *

☐ From the beginning offset

☒ From the latest offset

Enable immediately

☒

Consumer offset *

☐ From the beginning offset
 ☒ From the latest offset

Enable immediately

☒

Authentication ⓘ *

Password type

☒ PLAIN
 ☐ SCRAM-SHA-256

Username

function-2-user

Password

.....

Function Service does not store the username and password, which are only used for authentication after the trigger has been created.

Description

Please enter

Cancel Create

Test the overall flow

Log back into the ECS client instance, and run the producer bash script:

Code block

```
1 bin/kafka-console-producer.sh \
2   --broker-list <BROKER_ENDPOINT>:9092 \
3   --topic topic-a
4 >{"user":"alice","action":"login"}
```

To see the logs of the functions, go to **Release management** tab, click the release id which is currently online.

vefaas-function-2

node Node.js 20.x

V1

Version:Latest

Test

Release

Function configuration

Code

Trigger

Release management

Monitoring

Logging

Resource management

Release notes

Release notes

Version information

Q Search by release ID

Release ID	Status	Target version	Historical version	Description	Started/ended at	Actions
ulgbw520vmcxz2ng	Release complete	V1	Online revision	-	<div>Started at:</div> <div>May 13, 2025 10:00:54 PM</div> <div>Ended at:</div> <div>May 13, 2025 10:01:01 PM</div>	Roll back

Then, click **Logging**, and there should be a pop-up showing the live logs.

Function list > Function details > Release details

ulg520vmcxz2ng

Release complete

Roll back

Change history < V0 > V1 Release ratio V1-100% Start time: May 13, 2025 10:00:54 PM End time: May 13, 2025 10:01:01 PM Description -

V1 100% V0 0%

All instances

1

V1 instance

1

Instances

Q Search instance name

C

Instance name	Availability zone	IP	Status	Function revision	Started at	Actions
t3cr95xo-xud4sjobbus-559876dfb8-845t2	Availability zone 1b	10.0.3.5	Ready	V1 Online revision	May 13, 2025 10:00:58 PM	Logging Webshell Migrate

If everything works, you should see the logs in **vefaas-function-2** showing the message with an additional timestamp information in the message.

Loading in real time

t3cr95xo-g4dboz7x8h-5d6b56f788-bj5gh

Ready

Please enter keywordQ

Auto refresh

⏏

2025-05-13 22:09:55 [FaaS System] starting user function at port 8000

2025-05-13 22:09:55 [FaaS System] pod is not in stand by mode, try load user code directly

2025-05-13 22:09:55 [FaaS System] using function "handler" from script "index" as function.

2025-05-13 22:09:55 [FaaS System] node runtime listen on 8000

2025-05-13 22:09:55 [FaaS System] Detected runtime version: 1.2.1.

2025-05-13 22:09:55 [FaaS System] Function initialized successfully.

2025-05-13 22:10:17 received new request, request id: 4f841694-aabb-4e07-a4a6-0d49936de3f9

2025-05-13 22:10:17 Received message from topic-b: [Object: null prototype] {

2025-05-13 22:10:17 user: 'win',

2025-05-13 22:10:17 action: 'login',

2025-05-13 22:10:17 processedAt: '2025-05-13T14:10:12.488Z'

2025-05-13 22:10:17 }

Sample logs in **vefaas-function-1**:

2025-05-13 22:10:12 received new request, request id: 5206cc58-6c88-421a-920c-63514e4b0c40

2025-05-13 22:10:12 event: {

2025-05-13 22:10:12 id: '5206cc58-6c88-421a-920c-63514e4b0c40',

2025-05-13 22:10:12 time: '2025-05-13T14:10:12.486Z',

2025-05-13 22:10:12 type: 'faas.kafka.event',

2025-05-13 22:10:12 source: '/faas/event/kafka/zz6twimp',

2025-05-13 22:10:12 specversion: '1.0',

2025-05-13 22:10:12 datacontenttype: 'application/octet-stream',

2025-05-13 22:10:12 subject: undefined,

2025-05-13 22:10:12 datacontentencoding: undefined,

2025-05-13 22:10:12 dataschema: undefined,

2025-05-13 22:10:12 data_base64: 'eyJ1c2VyIjoibG9naW9uIjoibG9naW41fQ==',

2025-05-13 22:10:12 schemaurl: undefined,

2025-05-13 22:10:12 consumer group: 'topic-a_zz6twimp',

2025-05-13 22:10:12 key: '',

2025-05-13 22:10:12 maxretrinum: '5',

2025-05-13 22:10:12 offset: '4',

2025-05-13 22:10:12 partition: '1',

2025-05-13 22:10:12 retriesforbadstatusrequests: '0',

2025-05-13 22:10:12 topic: 'topic-a',

2025-05-13 22:10:12 eventType: 'cloudevent',

2025-05-13 22:10:12 isBase64Encoded: false,

2025-05-13 22:10:12 data: [Object: null prototype] { user: 'win', action: 'login' }

2025-05-13 22:10:12 }

2025-05-13 22:10:12 Successfully processed and sent this message to topic-b: "{"user":"win","action":"login","processedAt":"2025-05-13T14:10:12.488Z"}"

↑

↓