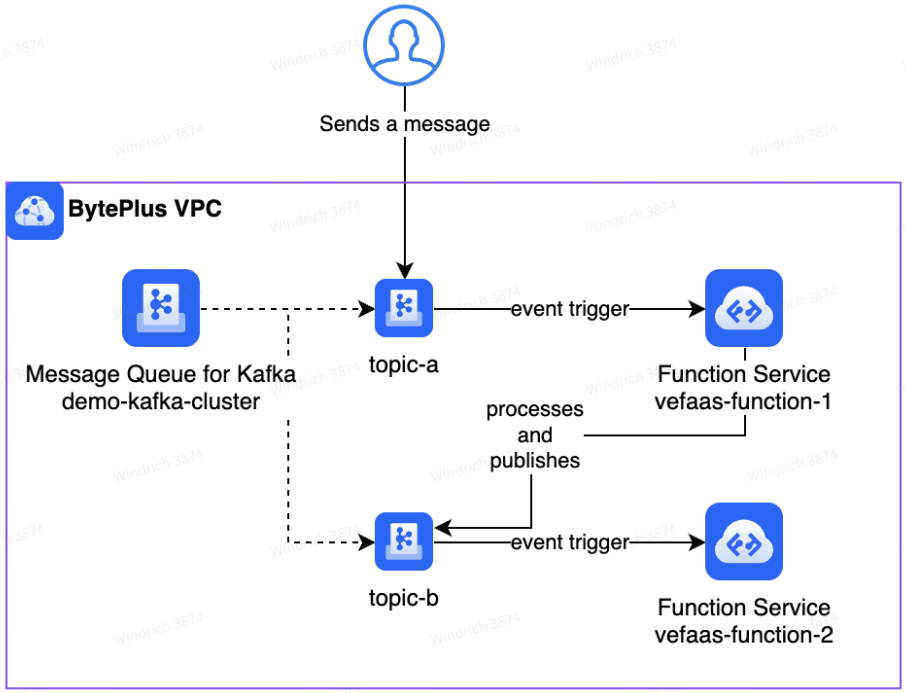
[External] BytePlus Function Service & Kafka Sample Deployment Guide

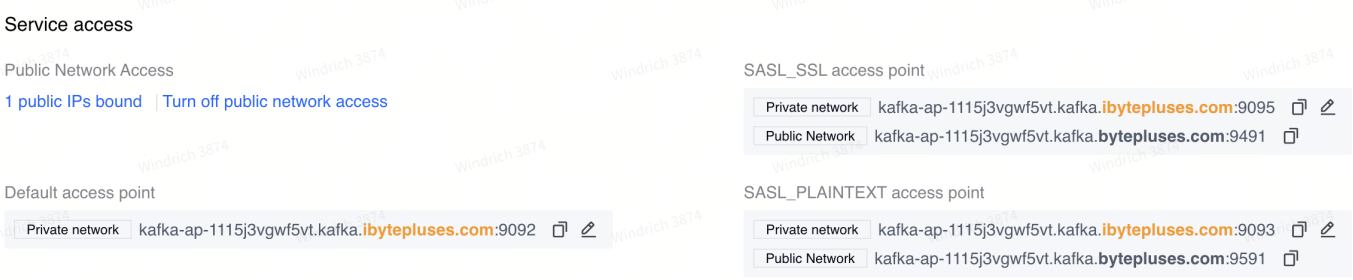
This deployment creates the necessary resources for the following flow:



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.

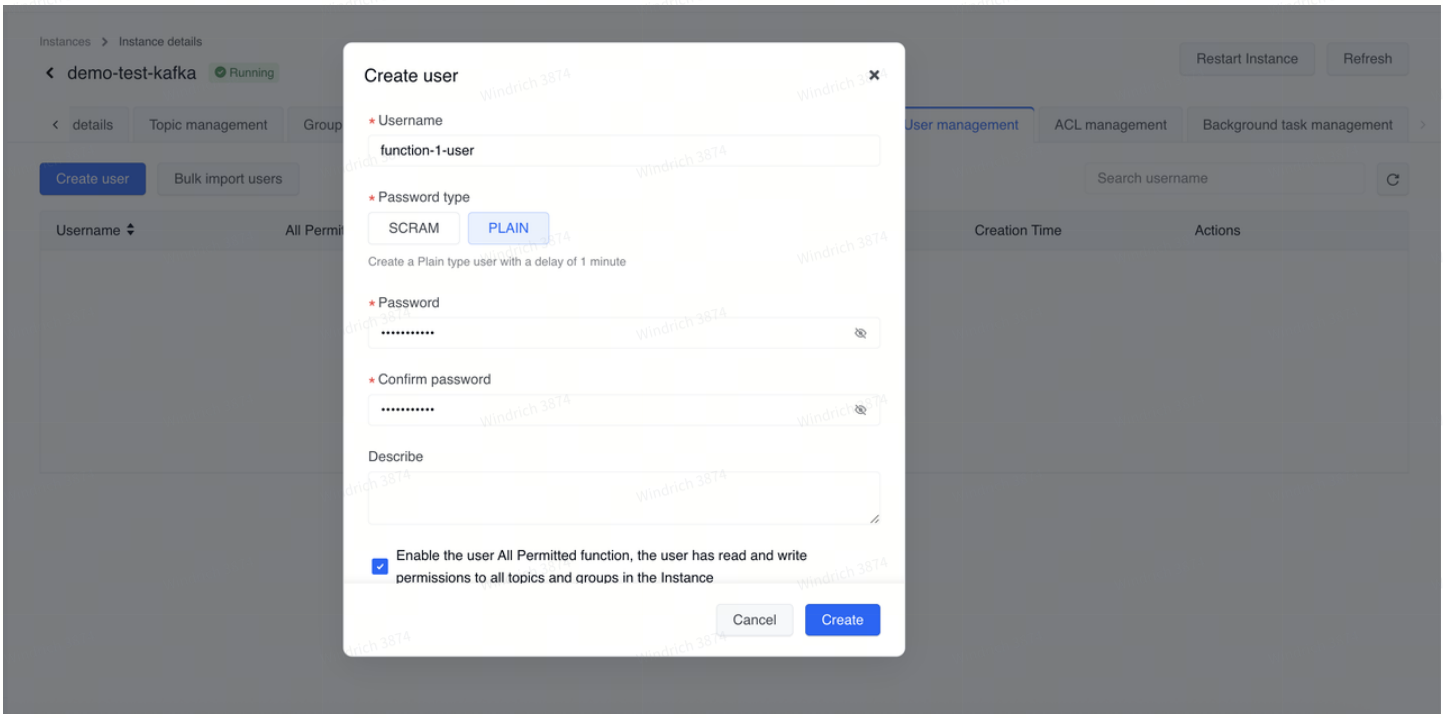


Create topics in the cluster

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

```
# Install Java
sudo apt update && sudo apt install -y openjdk-17-jre-headless

# Download and extract Kafka 3.7.2
wget https://downloads.apache.org/kafka/3.7.2/kafka_2.13-3.7.2.tgz
tar -xzf kafka_2.13-3.7.2.tgz
cd kafka_2.13-3.7.2
```

Start consuming messages from topic-a:

```
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

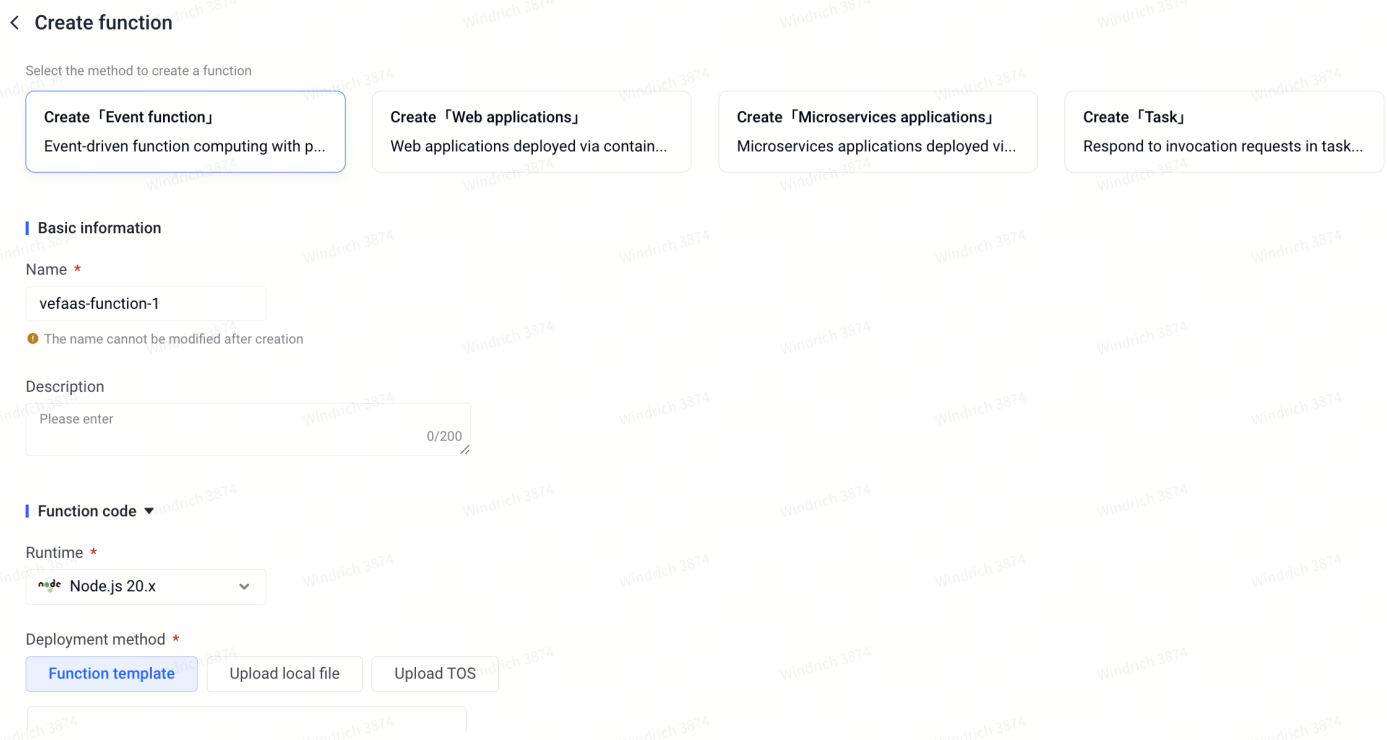
```
bin/kafka-console-producer.sh \
  --broker-list <BROKER_ENDPOINT>:9092 \
  --topic topic-a
>{"user":"alice","action":"login"}
```

The messages should appear on the consumer side.

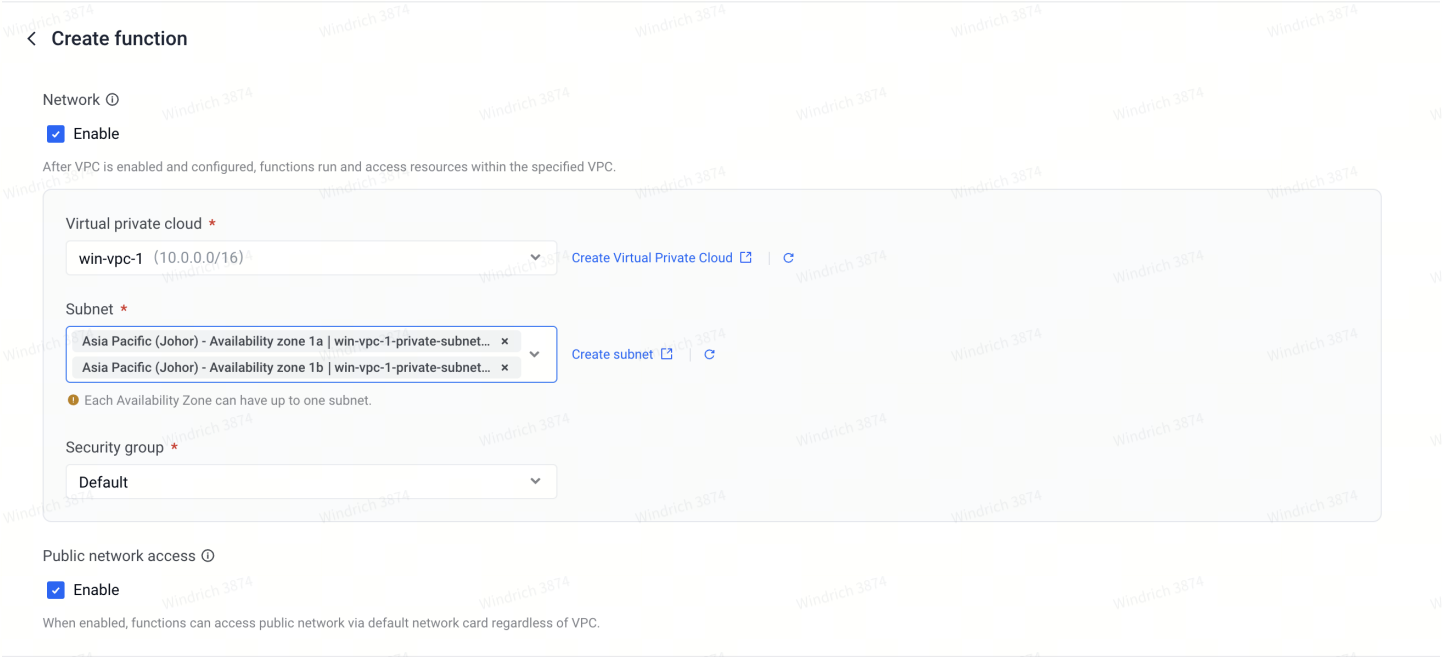
Create Serverless Functions

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`".



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.



Leave the rest to default and create the function.

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

Code block

```
{
  "name": "vefaas-function-1",
  "version": "1.0.0",
  "description": "Read and process message from a Kafka topic, and publish to another Kafka topic",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "dependencies": {
    "kafkajs": "^2.2.0"
  }
}
```

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

Code block

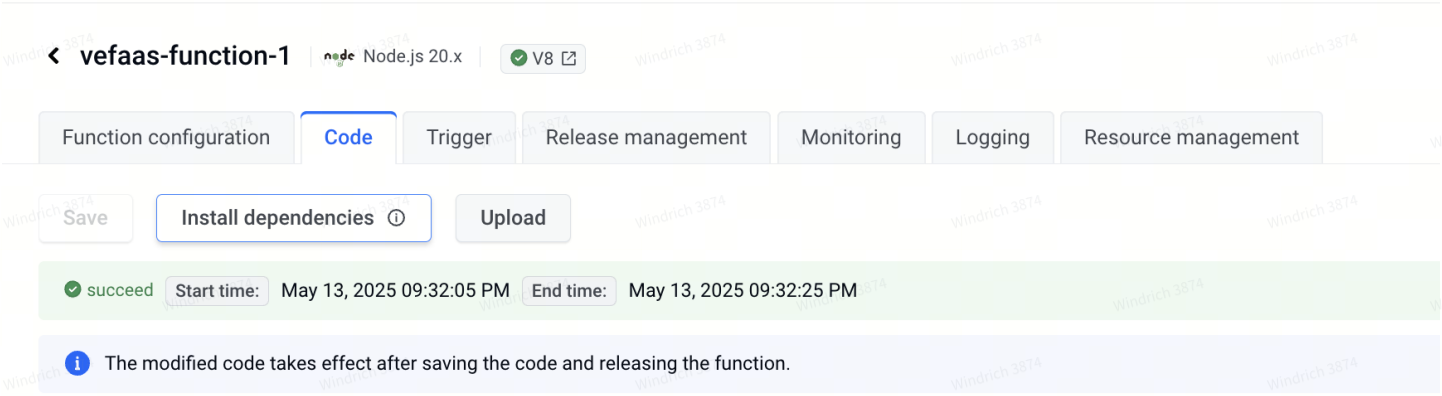
```
const { Kafka } = require('kafkajs');

const kafka = new Kafka({
  clientId: 'function1-producer',
  brokers: [process.env.KAFKA_BOOTSTRAP_SERVER],
  ssl: false, // Set to true if your Kafka cluster requires SSL
```

```
    sasl: {
      mechanism: 'plain',
      username: process.env.KAFKA_USERNAME, // Your Kafka username
      password: process.env.KAFKA_PASSWORD // Store password in environment variable
    }
});

exports.handler = async function handler(event, context) {
  const topicName = 'topic-b';

  console.log(`received new request, request id: %s`, context.requestId);
  console.log(`event: `, event);
  const message = event.data;

  // Simple processing: append a processedAt timestamp
  message.processedAt = new Date().toISOString();

  const producer = kafka.producer();
  await producer.connect();
  await producer.send({
    topic: topicName,
    messages: [{ value: JSON.stringify(message) }],
  });
  await producer.disconnect();

  console.log(`Successfully processed and sent this message to ${topicName}: "${JSON.stringify(message)}"`);

  return {
    statusCode: 200,
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ 'message': `message received and processed: ${JSON.stringify(message)}` }),
  };
};

```

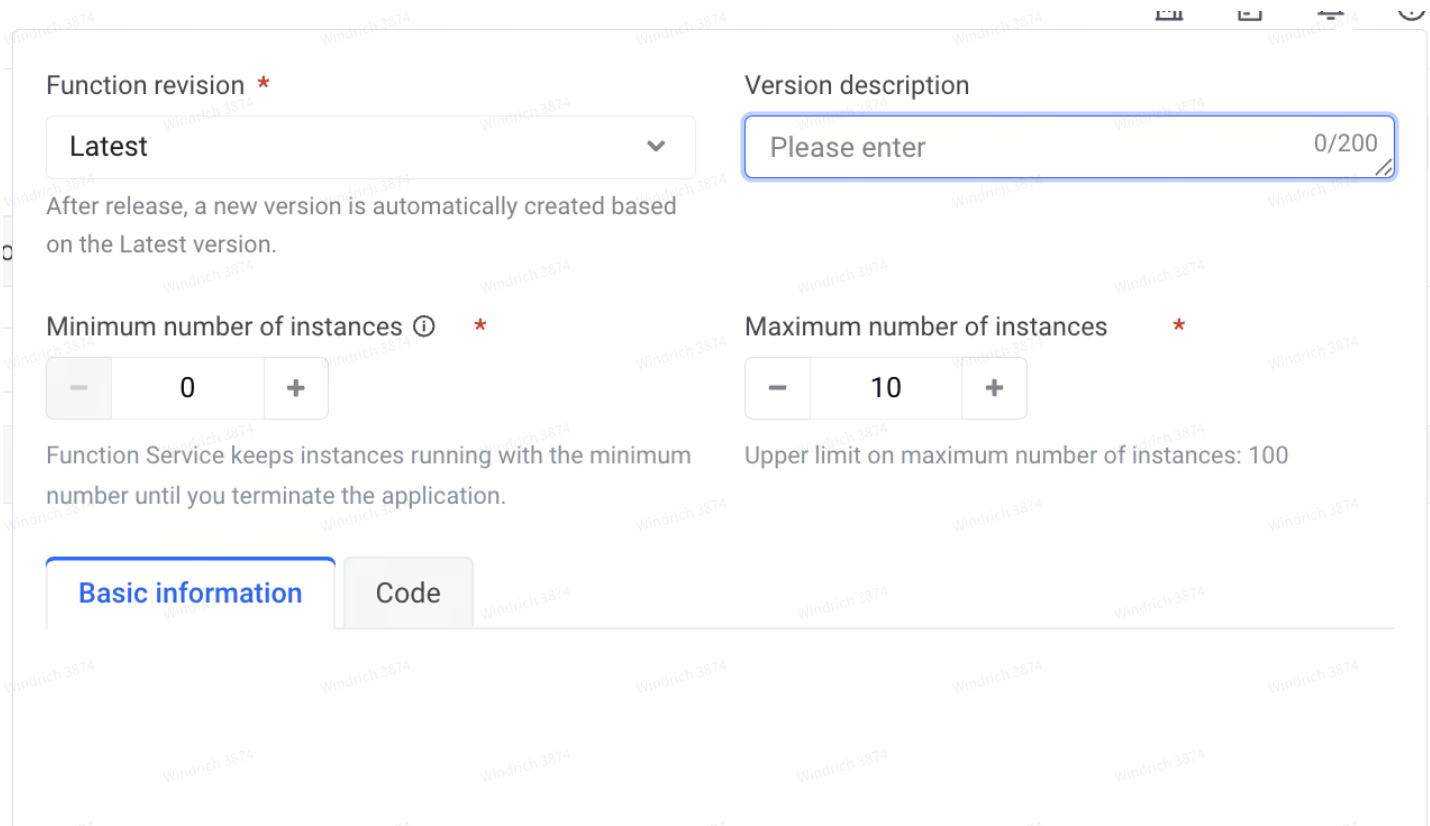
Go to function configuration and under Environment variables, add the variables:

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

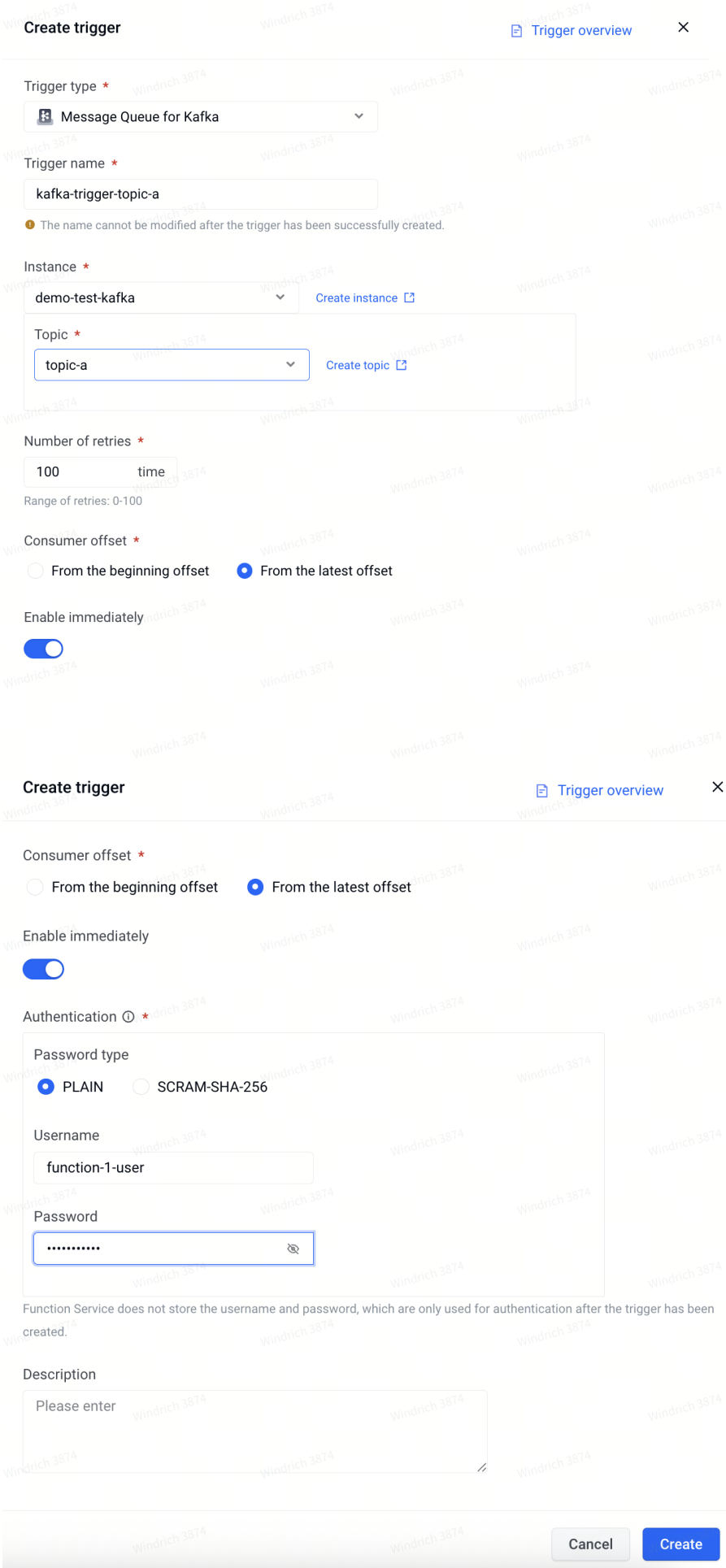
Under Code tab, install the required dependencies by clicking the `Install dependencies` button.



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 trigger
Trigger overview
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 trigger
Trigger overview
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
Cancel
Create

Create vefaas-function-2

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

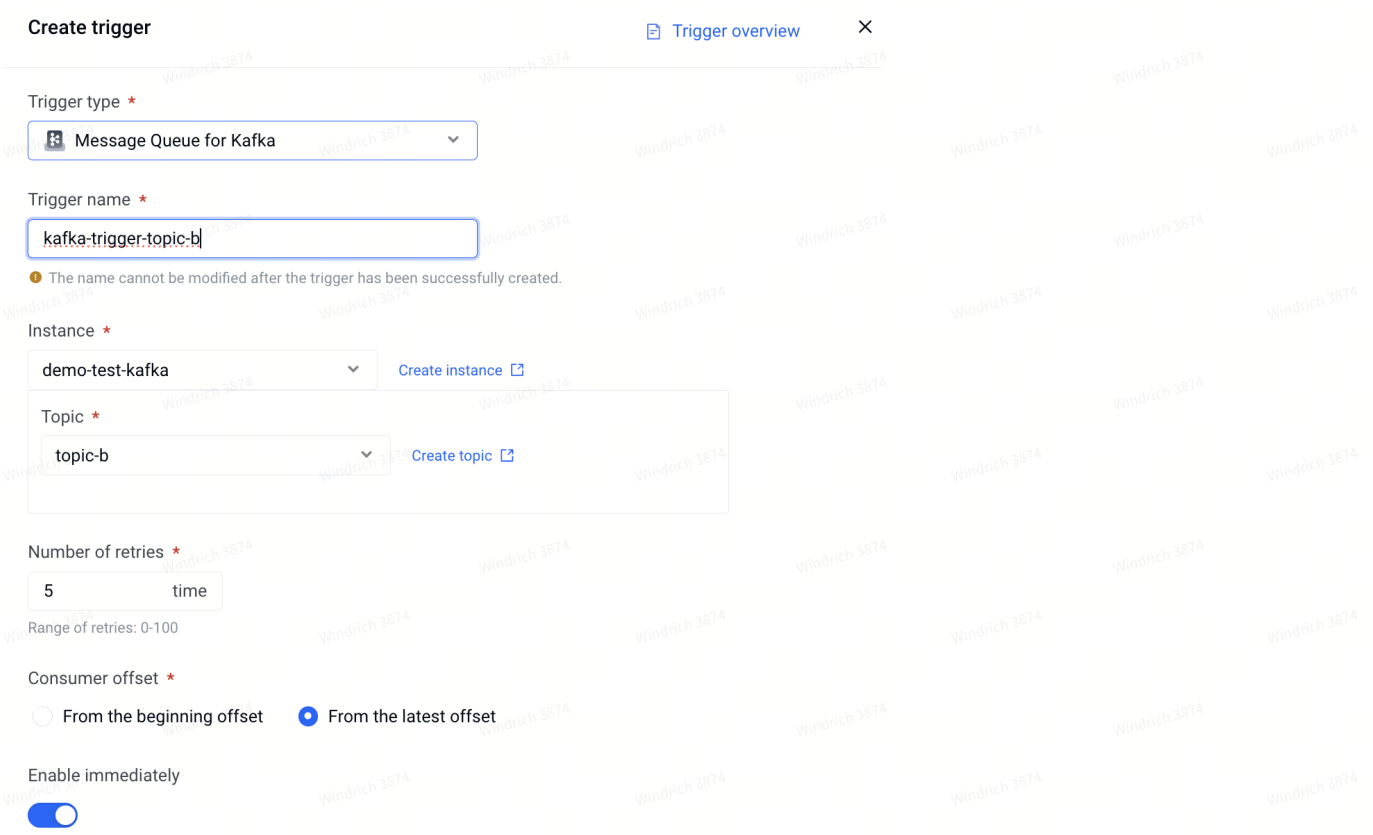
As this function simply prints out messages in `topic-b`, there's no need for additional dependencies or environment variables.

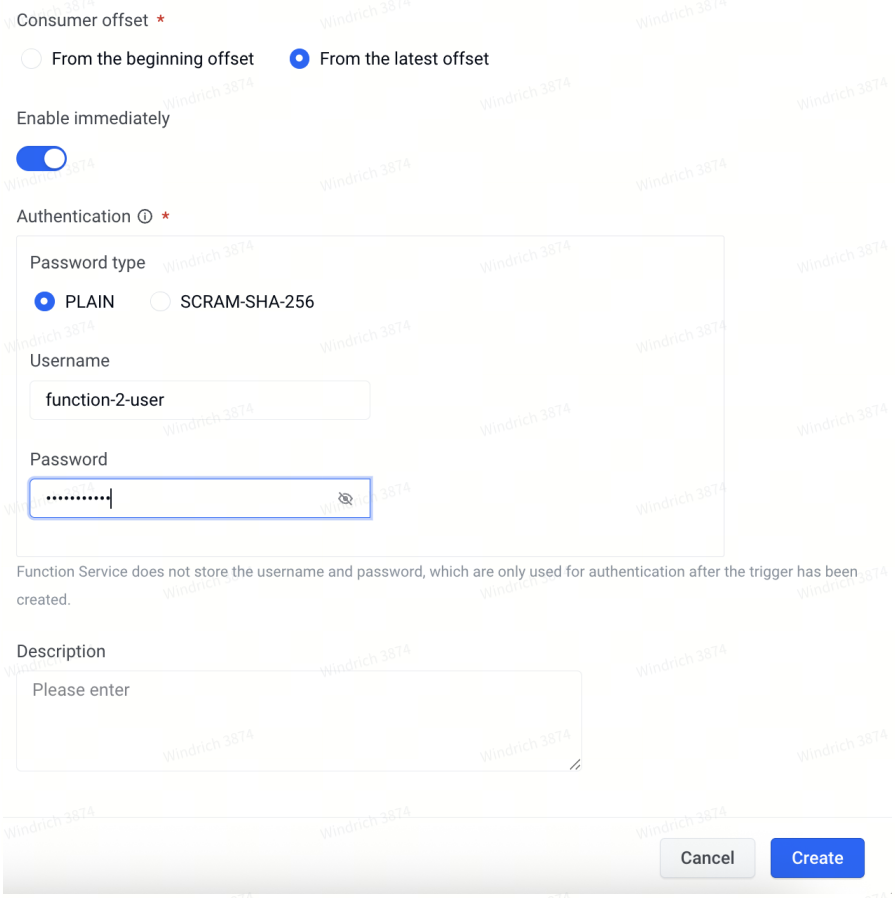
Add the following code for index.js, and then Release the function.

Code block

```
exports.handler = async function handler(event, context) {
  console.log(`received new request, request id: %s`, context.requestId);
  console.log(`Received message from topic-b: `, event.data);

  return {
    statusCode: 200,
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ 'message': `received message from topic-b: ${event.data}` }),
  };
};
```

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





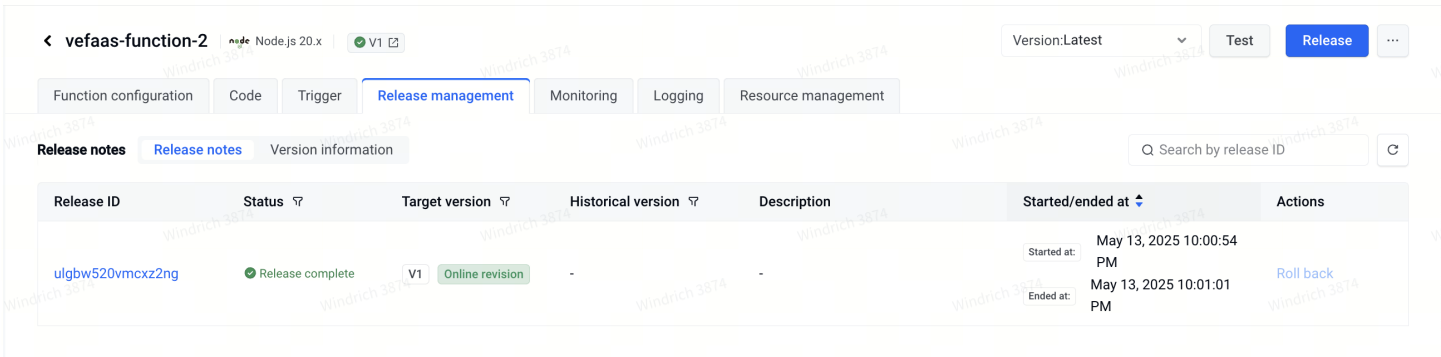
Test the overall flow

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

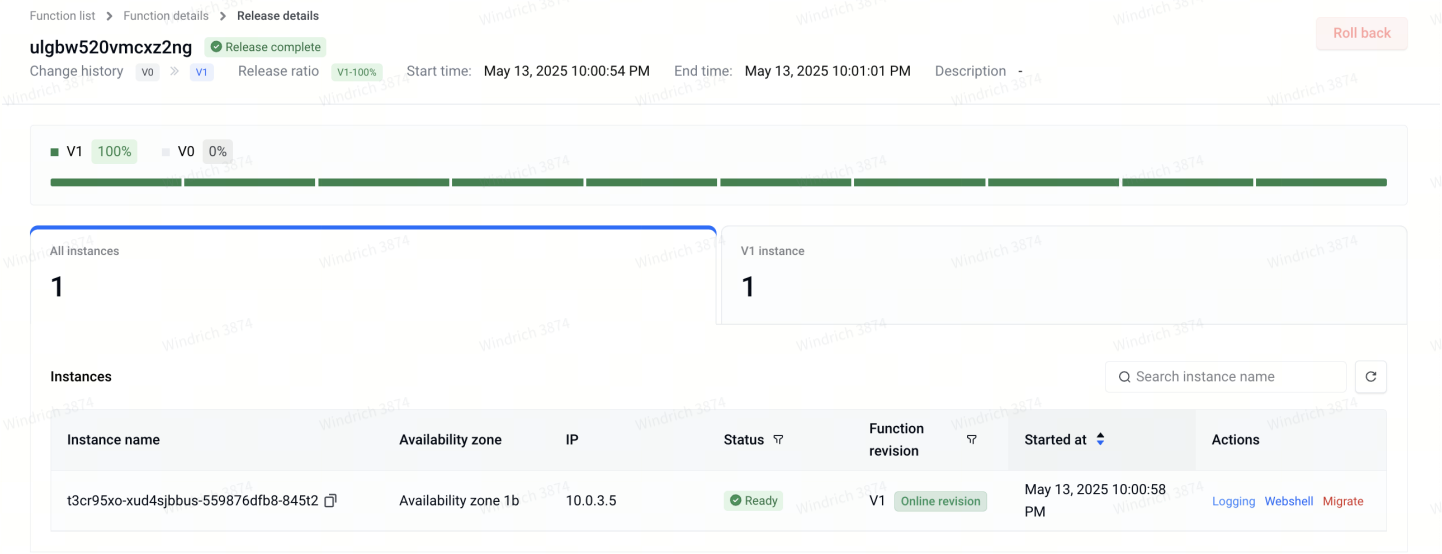
Code block

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

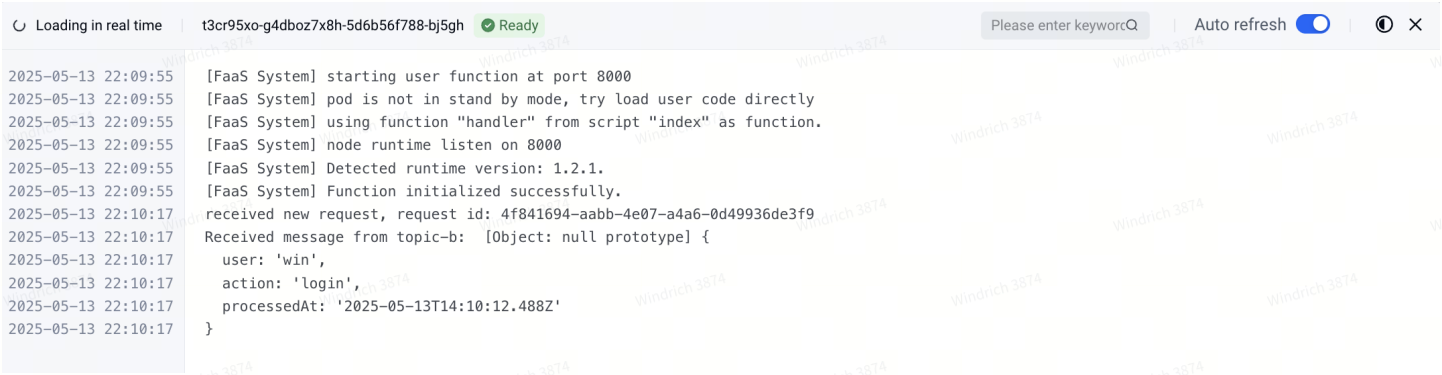
To see the logs of the functions, go to `Release management` tab, click the release id which is currently online.



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



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



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: 'eyJ1c2VyIjoid2luIiwiYWN0aW9uIjoibG9naW4ifQ==',
2025-05-13 22:10:12   schemaurl: undefined,
2025-05-13 22:10:12   consumergroup: 'topic-a_zz6twimp',
2025-05-13 22:10:12   key: '',
2025-05-13 22:10:12   maxretrynum: '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"}"
```