[#tokens-number] == Working with number-encoded tokens

Number-encoded tokens are a set of tiny negative floating-point numbers that look like the following.

== [source,none,subs=“verbatim,attributes”]

## -1.8881545897087626e+289

As with list tokens, you cannot modify the number value, as doing so is likely to break the number token.

The following is an example of a construct that contains a token encoded as a number:

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import { Stack, Duration, StackProps } from ‘aws-cdk-lib’; import { Construct } from ‘constructs’; import \* as rds from ‘aws-cdk-lib/aws-rds’; import \* as ec2 from ‘aws-cdk-lib/aws-ec2’;

export class CdkDemoAppStack extends Stack { constructor(scope: Construct, id: string, props?: StackProps) { super(scope, id, props);

…. // Define a new VPC const vpc = new ec2.Vpc(this, ‘MyVpc’, { maxAzs: 3, // Maximum number of availability zones to use });

// Define an RDS database cluster const dbCluster = new rds.DatabaseCluster(this, ‘MyRDSCluster’, { engine: rds.DatabaseClusterEngine.AURORA, instanceProps: { vpc, }, });

// Get the port token (this is a token encoded as a number) const portToken = dbCluster.clusterEndpoint.port;

// Print the value for our token at synthesis console.log(“portToken:” + portToken); } } —- ….

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const { Stack, Duration } = require(‘aws-cdk-lib’); const lambda = require(‘aws-cdk-lib/aws-lambda’); const rds = require(‘aws-cdk-lib/aws-rds’); const ec2 = require(‘aws-cdk-lib/aws-ec2’);

class CdkDemoAppStack extends Stack { constructor(scope, id, props) { super(scope, id, props);

…. // Define a new VPC const vpc = new ec2.Vpc(this, ‘MyVpc’, { maxAzs: 3, // Maximum number of availability zones to use });

// Define an RDS database cluster const dbCluster = new rds.DatabaseCluster(this, ‘MyRDSCluster’, { engine: rds.DatabaseClusterEngine.AURORA, instanceProps: { vpc, }, });

// Get the port token (this is a token encoded as a number) const portToken = dbCluster.clusterEndpoint.port;

// Print the value for our token at synthesis console.log(“portToken:” + portToken); } } ….

== module.exports = { CdkDemoAppStack }

Python:: + [source,python,subs=“verbatim,attributes”] — from aws\_cdk import ( Duration, Stack, ) from aws\_cdk import aws\_rds as rds from aws\_cdk import aws\_ec2 as ec2 from constructs import Construct

class CdkDemoAppStack(Stack):

…. def **init**(self, scope: Construct, construct\_id: str, \*\*kwargs) -> None: super().\_\_init\_\_(scope, construct\_id, \*\*kwargs)

# Define a new VPC  
vpc = ec2.Vpc(self, 'MyVpc',  
 max\_azs=3 # Maximum number of availability zones to use  
)  
  
# Define an RDS database cluster  
db\_cluster = rds.DatabaseCluster(self, 'MyRDSCluster',  
 engine=rds.DatabaseClusterEngine.AURORA,  
 instance\_props=rds.InstanceProps(  
 vpc=vpc  
 )  
)  
  
# Get the port token (this is a token encoded as a number)  
port\_token = db\_cluster.cluster\_endpoint.port  
  
# Print the value for our token at synthesis  
print(f"portToken: {port\_token}") ----

….

Java:: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

import software.constructs.Construct; import software.amazon.awscdk.Stack; import software.amazon.awscdk.StackProps; import software.amazon.awscdk.services.ec2.Vpc; import software.amazon.awscdk.services.rds.DatabaseCluster; import software.amazon.awscdk.services.rds.DatabaseClusterEngine; import software.amazon.awscdk.services.rds.InstanceProps;

public class CdkDemoAppStack extends Stack { public CdkDemoAppStack(final Construct scope, final String id) { this(scope, id, null); }

…. public CdkDemoAppStack(final Construct scope, final String id, final StackProps props) { super(scope, id, props);

// Define a new VPC  
Vpc vpc = Vpc.Builder.create(this, "MyVpc")  
 .maxAzs(3) // Maximum number of availability zones to use  
 .build();  
  
// Define an RDS database cluster  
DatabaseCluster dbCluster = DatabaseCluster.Builder.create(this, "MyRDSCluster")  
 .engine(DatabaseClusterEngine.AURORA)  
 .instanceProps(InstanceProps.builder()  
 .vpc(vpc)  
 .build())  
 .build();  
  
// Get the port token (this is a token encoded as a number)  
Number portToken = dbCluster.getClusterEndpoint().getPort();  
  
// Print the value for our token at synthesis  
System.out.println("portToken: " + portToken);

} } —- ….

C#:: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using Constructs; using Amazon.CDK.{aws}.EC2; using Amazon.CDK.{aws}.RDS; using System; using System.Collections.Generic;

namespace CdkDemoApp { public class CdkDemoAppStack : Stack { internal CdkDemoAppStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props) { // Define a new VPC var vpc = new Vpc(this, “MyVpc”, new VpcProps { MaxAzs = 3 // Maximum number of availability zones to use });

…. // Define an RDS database cluster var dbCluster = new DatabaseCluster(this, “MyRDSCluster”, new DatabaseClusterProps { Engine = DatabaseClusterEngine.AURORA, // Remove parentheses InstanceProps = new Amazon.CDK.{aws}.RDS.InstanceProps // Specify RDS InstanceProps { Vpc = vpc } });

// Get the port token (this is a token encoded as a number)  
 var portToken = dbCluster.ClusterEndpoint.Port;  
  
 // Print the value for our token at synthesis  
 System.Console.WriteLine($"portToken: {portToken}");  
}

} } —- ….

Go:: + [source,go,subs=“verbatim,attributes”] — package main

import ( “fmt”

“github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/aws-cdk-go/awscdk/v2/awsec2” “github.com/aws/aws-cdk-go/awscdk/v2/awsrds” “github.com/aws/constructs-go/constructs/v10” “github.com/aws/jsii-runtime-go” )

type CdkDemoAppStackProps struct { awscdk.StackProps }

func NewCdkDemoAppStack(scope constructs.Construct, id string, props \*CdkDemoAppStackProps) awscdk.Stack { var sprops awscdk.StackProps if props != nil { sprops = props.StackProps } stack := awscdk.NewStack(scope, &id, &sprops)

…. // Define a new VPC vpc := awsec2.NewVpc(stack, jsii.String(“MyVpc”), &awsec2.VpcProps{ MaxAzs: jsii.Number(3), // Maximum number of availability zones to use })

// Define an RDS database cluster dbCluster := awsrds.NewDatabaseCluster(stack, jsii.String(“MyRDSCluster”), &awsrds.DatabaseClusterProps{ Engine: awsrds.DatabaseClusterEngine\_AURORA(), InstanceProps: &awsrds.InstanceProps{ Vpc: vpc, }, })

// Get the port token (this is a token encoded as a number) portToken := dbCluster.ClusterEndpoint().Port()

// Print the value for our token at synthesis fmt.Println(“portToken:”, portToken)

return stack } ….

== // …

====

When we run cdk synth, the value for portToken is displayed as a number-encoded token:

== [source,bash,subs=“verbatim,attributes”]

$ cdk synth –quiet portToken: -1.8881545897087968e+289 —