Serverless Stack (SST) Framework

Introduction

What is Serverless?

In the context of web development, "serverless" refers to a model where the server-side logic for an application is provided by a cloud service provider, rather than running on a dedicated server

In a serverless architecture, the developer writes code to handle specific functions or events, and the cloud provider automatically provisions the necessary compute resources to run that code, scaling up or down as needed to handle traffic and usage. This means that the developer does not need to worry about managing servers, infrastructure, or scaling - the cloud provider takes care of all of that.

Despite the name, serverless does not mean that there are no servers involved at all. Rather, the term "serverless" refers to the idea that the developer does not need to manage or think about the servers, as they are abstracted away by the cloud provider.

Serverless Computing

Serverless computing (or serverless for short), is an execution model where the cloud provider (AWS, Azure, or Google Cloud) is responsible for executing a piece of code by dynamically allocating the resources. And only charging for the amount of resources used to run the code. The code is typically run inside stateless containers that can be triggered by a variety of events including http requests, database events, queuing services, monitoring alerts, file uploads, scheduled events (cron jobs), etc. The code that is sent to the cloud provider for execution is usually in the form of a function. Hence serverless is sometimes referred to as *“Functions as a Service”* or *“FaaS”*. Following are the FaaS offerings of the major cloud providers:

- AWS: AWS Lambda
- Microsoft Azure: Azure Functions
- Google Cloud: Cloud Functions

While serverless abstracts the underlying infrastructure away from the developer, servers are still involved in executing our functions.

Before Serverless

Traditionally, we’ve built and deployed web applications where we have some degree of control over the HTTP requests that are made to our server. Our application runs on that server and we are responsible for provisioning and managing the resources for it. There are a few issues with this.

1. We are charged for keeping the server up even when we are not serving out any requests.
2. We are responsible for uptime and maintenance of the server and all its resources.
3. We are also responsible for applying the appropriate security updates to the server.
4. As our usage scales we need to manage scaling up our server as well. And as a result manage scaling it down when we don’t have as much usage.

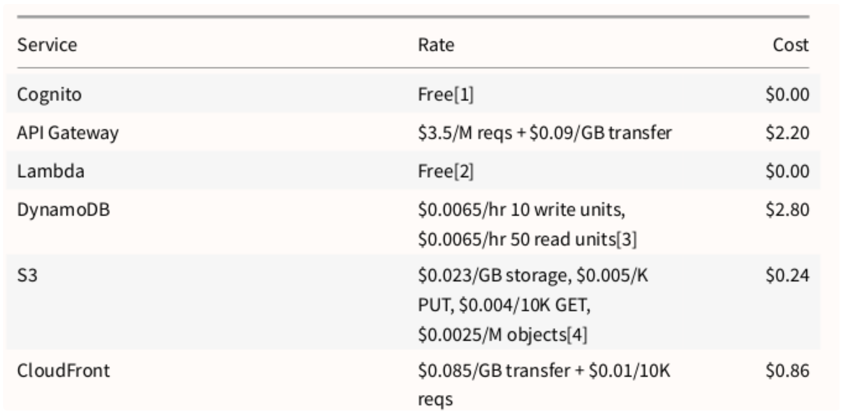
For smaller companies and individual developers this can be a lot to handle. This ends up distracting from the more important job that we have; building and maintaining the actual application. At larger organizations this is handled by the infrastructure team and usually it is not the responsibility of the individual developer. However, the processes necessary to support this can end up slowing down development times. As you cannot just go ahead and build your application without working with the infrastructure team to help you get up and running. As developers we’ve been looking for a solution to these problems and this is where serverless comes in.

Why Create Serverless Apps?

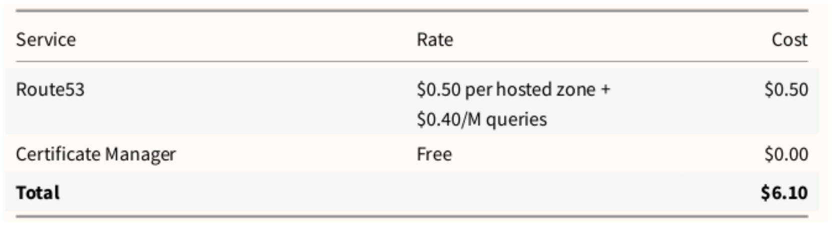
It is important to address why it is worth learning how to create serverless apps. There are a few reasons why serverless apps are favored over traditional server hosted apps:

1. Low maintenance
2. Low cost
3. Easy to scale

The biggest benefit by far is that you only need to worry about your code and nothing else. The low maintenance is a result of not having any servers to manage. You don't need to actively ensure that your server is running properly, or that you have the right security updates on it. You deal with your own application code and nothing else.
The main reason it's cheaper to run serverless applications is that you are effectively only paying per request. So when your application is not being used, you are not being charged for it. Let's do a quick breakdown of what it would cost for us to run our note taking application. We'll assume that we have 1000 daily active users making 20 requests per



Service	Rate	Cost
Cognito	Free[1]	$0.00
API Gateway	$3.5/M reqs + $0.09/GB transfer	$2.20
Lambda	Free[2]	$0.00
DynamoDB	$0.0065/hr 10 write units, $0.0065/hr 50 read units[3]	$2.80
S3	$0.023/GB storage, $0.005/K PUT, $0.004/10K GET, $0.0025/M objects[4]	$0.24
CloudFront	$0.085/GB transfer + $0.01/10K reqs	$0.86



Service	Rate	Cost
Route53	$0.50 per hosted zone + $0.40/M queries	$0.50
Certificate Manager	Free	$0.00
Total		**$6.10**

- Cognito is free for < 50K MAUs and $0.00550/MAU onwards.
- Lambda is free for < 1M requests and 400000GB-secs of compute.
- DynamoDB gives 25GB of free storage.
- S3 gives 1GB of free transfer.

So that comes out to $6.10 per month. Additionally, a .com domain would cost us $12 per year, making that the biggest up front cost for us. But just keep in mind that these are very rough estimates. Real-world usage patterns are going to be very different. However, these rates should give you a sense of how the cost of running a serverless application is calculated.
Finally, the ease of scaling is thanks in part to DynamoDB which gives us near infinite scale and Lambda that simply scales up to meet the demand. And of course our front end is a simple static single page app that is almost guaranteed to always respond instantly thanks to CloudFront.

Deploy your first Next.Js app to AWS with SST

Prerequisites

You'll need at least Node.js 16 and npm 7. You also need to have an AWS account and **AWS credentials configured locally**.

1. configure was credentilas

```
aws configure
AWS Access Key ID [****************LKHE]: AKIAAUHEIRRDDCFEES
AWS Secret Access Key [****************edeo]: endeneemjdofifjrjfniue
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

2. Create a new NextJs app

To create the app you can use yarn, npm or pnpm package mangaer

```
yarn create next-app
```

It will create a simple NextJS app locally

3. Now initialize SST in your project root.

```
yarn create sst
```

It will Initialize SST in your NextJs project by adding SST script in package.json file, sst.config.ts which contains the Infrastructure code. SST uses AWS CDK to create the infrastructure.

4. Now install yarn in your project root.

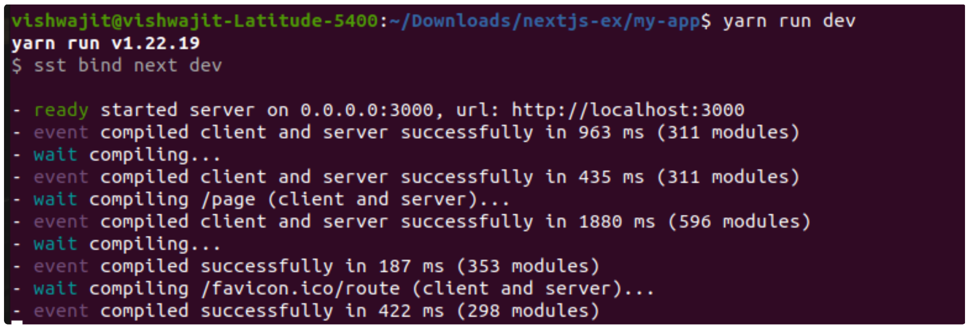
```
yarn install
```

It will Initialize SST in your NextJs project by adding SST script in package.json file, sst.config.ts which contains the Infrastructure code. SST uses AWS CDK to create the infrastructure.

5. Test NextJs app locally first

```
yarn run dev
```

It will start your nextjs app locally and get you localhost URL and if everything looks ok you can deploy your app in final stage which is prod or any other stage according to your requirement

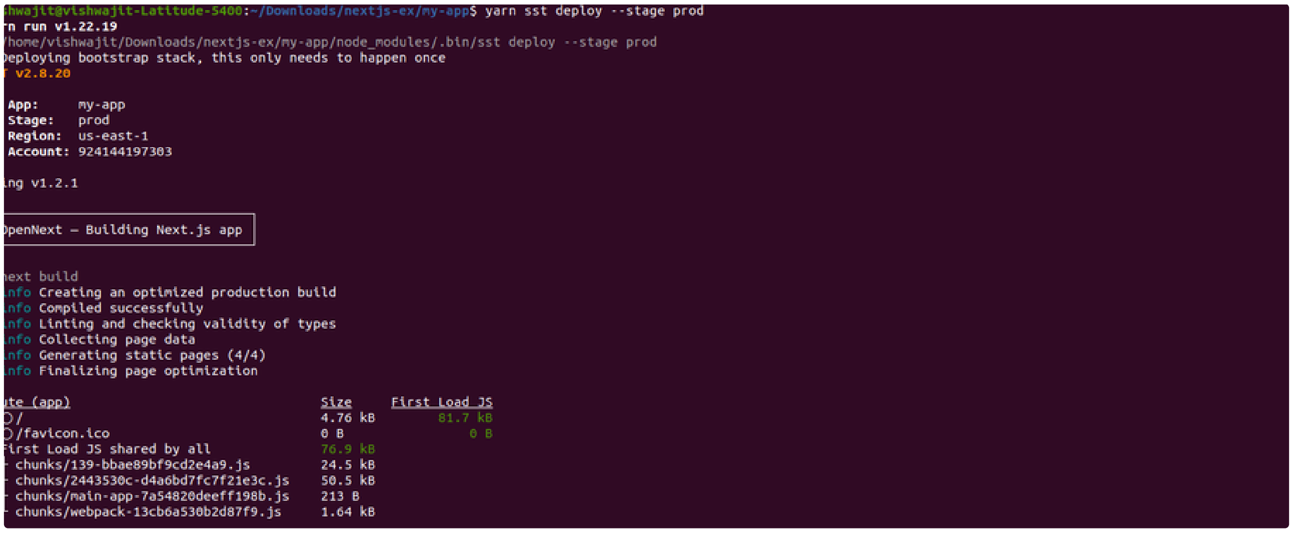


```
vishwajit@vishwajit-Latitude-5400:~/Downloads/nextjs-ex/my-app$ yarn run dev
yarn run v1.22.19
$ sst bind next dev

- ready started server on 0.0.0.0:3000, url: http://localhost:3000
- event compiled client and server successfully in 963 ms (311 modules)
- wait compiling...
- event compiled client and server successfully in 435 ms (311 modules)
- wait compiling /page (client and server)...
- event compiled client and server successfully in 1880 ms (596 modules)
- wait compiling...
- event compiled successfully in 187 ms (353 modules)
- wait compiling /favicon.ico/route (client and server)...
- event compiled successfully in 422 ms (298 modules)
```

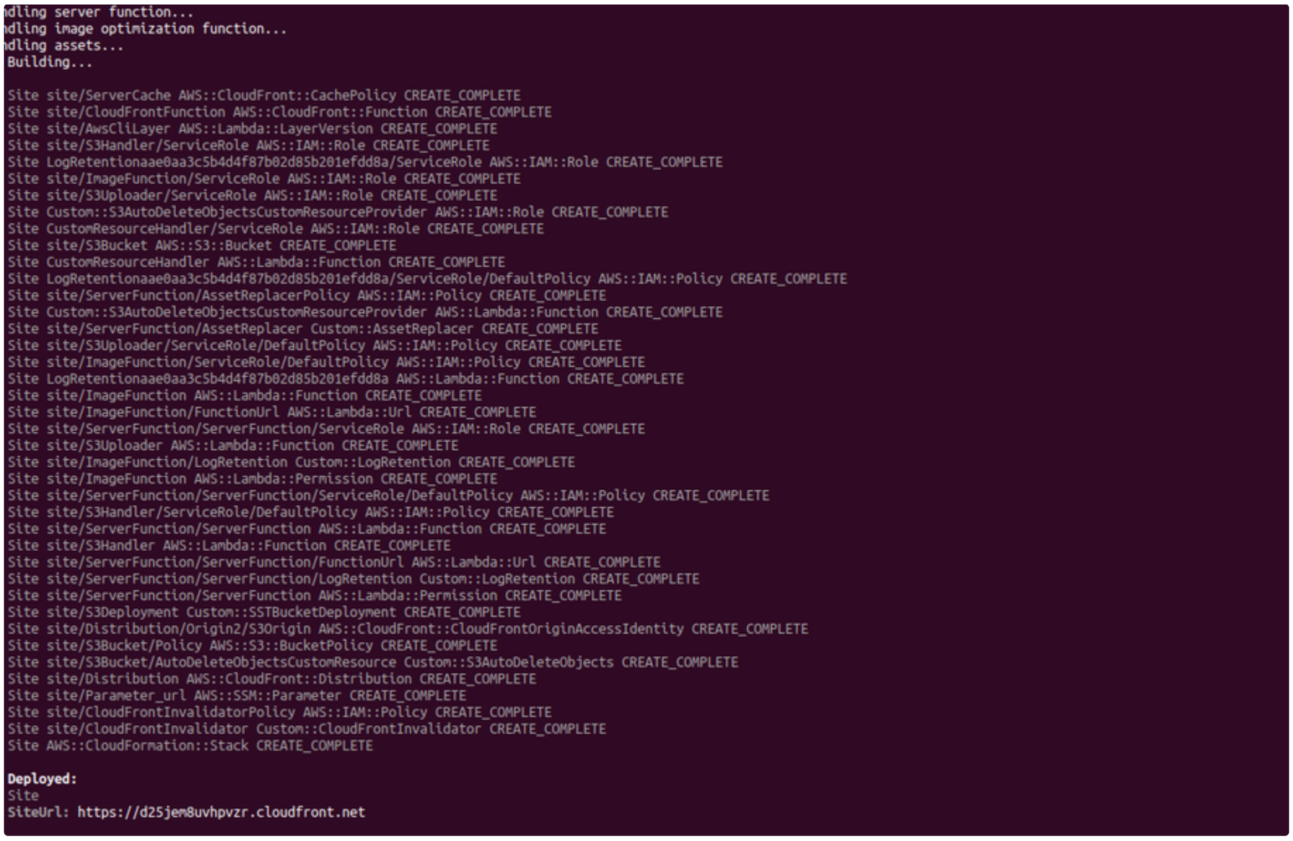
6. Now Deploy Your app

```
yarn sst deploy --stage prod
```

Now deploy your app with specific stage as per your requirement when you run following command then you will get API Endpoint URL or SiteUrl

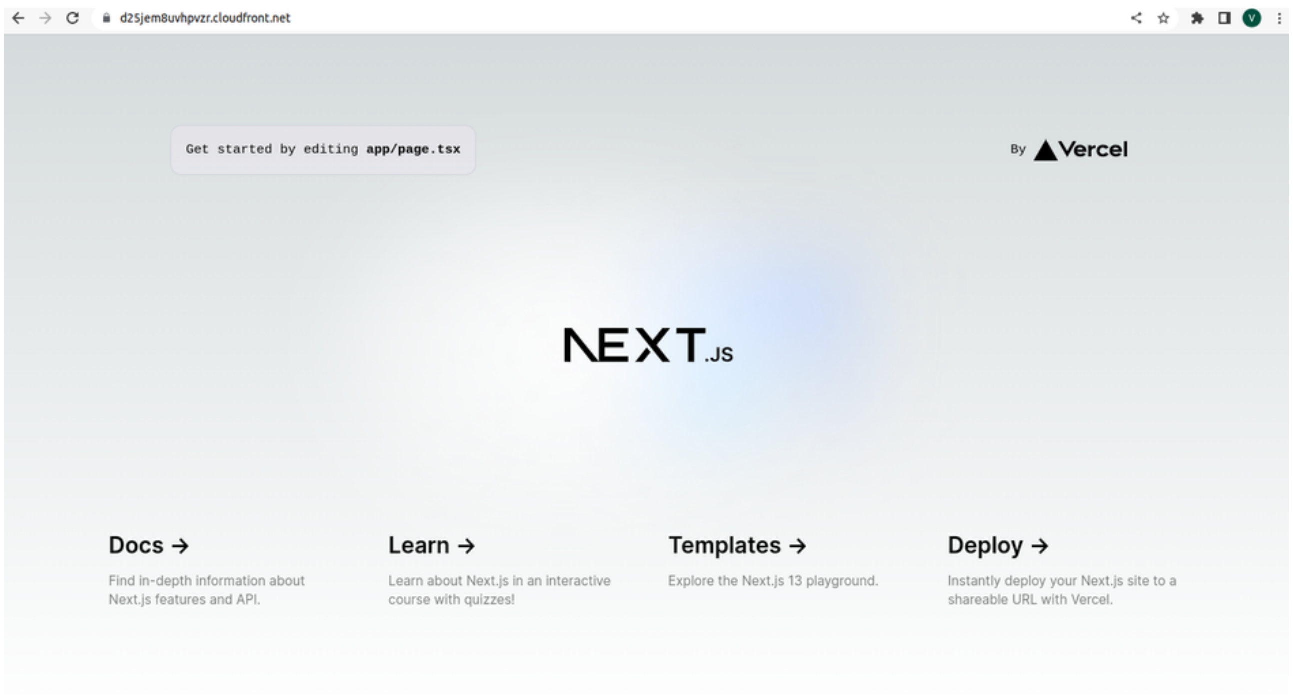


```
hwajit@vishwajit-Latitude-5400:~/Downloads/nextjs-ex/my-app$ yarn sst deploy --stage prod
n run v1.22.19
/home/vishwajit/Downloads/nextjs-ex/my-app/node_modules/.bin/sst deploy --stage prod
eploying bootstrap stack, this only needs to happen once
 v2.8.20

 App:     my-app
 Stage:   prod
 Region:  us-east-1
 Account: 924144197303

ng v1.2.1

penNext — Building Next.js app

ext build
nfo Creating an optimized production build
nfo Compiled successfully
nfo Linting and checking validity of types
nfo Collecting page data
nfo Generating static pages (4/4)
nfo Finalizing page optimization

te (app)                              Size     First Load JS
)/                                    4.76 kB        81.7 kB
)/favicon.ico                         0 B               0 B
irst Load JS shared by all            76.9 kB
 chunks/139-bbae89bf9cd2e4a9.js       24.5 kB
 chunks/2443530c-d4a6bd7fc7f21e3c.js  50.5 kB
 chunks/main-app-7a54820deeff198b.js  213 B
 chunks/webpack-13cb6a530b2d87f9.js   1.64 kB
```

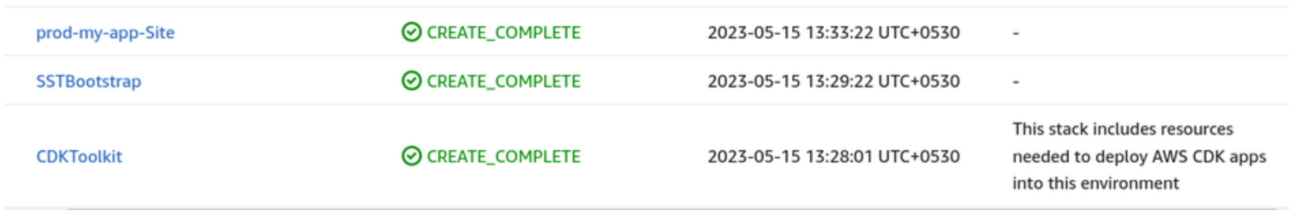


```
dling server function...
dling image optimization function...
dling assets...
 Building...

 Site site/ServerCache AWS::CloudFront::CachePolicy CREATE_COMPLETE
 Site site/CloudFrontFunction AWS::CloudFront::Function CREATE_COMPLETE
 Site site/AwsCliLayer AWS::Lambda::LayerVersion CREATE_COMPLETE
 Site site/S3Handler/ServiceRole AWS::IAM::Role CREATE_COMPLETE
 Site LogRetentionaae0aa3c5b4d4f87b02d85b201efdd8a/ServiceRole AWS::IAM::Role CREATE_COMPLETE
 Site site/ImageFunction/ServiceRole AWS::IAM::Role CREATE_COMPLETE
 Site site/S3Uploader/ServiceRole AWS::IAM::Role CREATE_COMPLETE
 Site Custom::S3AutoDeleteObjectsCustomResourceProvider AWS::IAM::Role CREATE_COMPLETE
 Site CustomResourceHandler/ServiceRole AWS::IAM::Role CREATE_COMPLETE
 Site site/S3Bucket AWS::S3::Bucket CREATE_COMPLETE
 Site CustomResourceHandler AWS::Lambda::Function CREATE_COMPLETE
 Site LogRetentionaae0aa3c5b4d4f87b02d85b201efdd8a/ServiceRole/DefaultPolicy AWS::IAM::Policy CREATE_COMPLETE
 Site site/ServerFunction/AssetReplacerPolicy AWS::IAM::Policy CREATE_COMPLETE
 Site Custom::S3AutoDeleteObjectsCustomResourceProvider AWS::Lambda::Function CREATE_COMPLETE
 Site site/ServerFunction/AssetReplacer Custom::AssetReplacer CREATE_COMPLETE
 Site site/S3Uploader/ServiceRole/DefaultPolicy AWS::IAM::Policy CREATE_COMPLETE
 Site site/ImageFunction/ServiceRole/DefaultPolicy AWS::IAM::Policy CREATE_COMPLETE
 Site LogRetentionaae0aa3c5b4d4f87b02d85b201efdd8a AWS::Lambda::Function CREATE_COMPLETE
 Site site/ImageFunction AWS::Lambda::Function CREATE_COMPLETE
 Site site/ImageFunction/FunctionUrl AWS::Lambda::Url CREATE_COMPLETE
 Site site/ServerFunction/ServerFunction/ServiceRole AWS::IAM::Role CREATE_COMPLETE
 Site site/S3Uploader AWS::Lambda::Function CREATE_COMPLETE
 Site site/ImageFunction/LogRetention Custom::LogRetention CREATE_COMPLETE
 Site site/ImageFunction AWS::Lambda::Permission CREATE_COMPLETE
 Site site/ServerFunction/ServerFunction/ServiceRole/DefaultPolicy AWS::IAM::Policy CREATE_COMPLETE
 Site site/S3Handler/ServiceRole/DefaultPolicy AWS::IAM::Policy CREATE_COMPLETE
 Site site/ServerFunction/ServerFunction AWS::Lambda::Function CREATE_COMPLETE
 Site site/S3Handler AWS::Lambda::Function CREATE_COMPLETE
 Site site/ServerFunction/ServerFunction/FunctionUrl AWS::Lambda::Url CREATE_COMPLETE
 Site site/ServerFunction/ServerFunction/LogRetention Custom::LogRetention CREATE_COMPLETE
 Site site/ServerFunction/ServerFunction AWS::Lambda::Permission CREATE_COMPLETE
 Site site/S3Deployment Custom::SSTBucketDeployment CREATE_COMPLETE
 Site site/Distribution/Origin2/S3Origin AWS::CloudFront::CloudFrontOriginAccessIdentity CREATE_COMPLETE
 Site site/S3Bucket/Policy AWS::S3::BucketPolicy CREATE_COMPLETE
 Site site/S3Bucket/AutoDeleteObjectsCustomResource Custom::S3AutoDeleteObjects CREATE_COMPLETE
 Site site/Distribution AWS::CloudFront::Distribution CREATE_COMPLETE
 Site site/Parameter_url AWS::SSM::Parameter CREATE_COMPLETE
 Site site/CloudFrontInvalidatorPolicy AWS::IAM::Policy CREATE_COMPLETE
 Site site/CloudFrontInvalidator Custom::CloudFrontInvalidator CREATE_COMPLETE
 Site AWS::CloudFormation::Stack CREATE_COMPLETE

 Deployed:
 Site
 SiteUrl: https://d25jem8uvhpvzr.cloudfront.net
```

It will create Infrastructure on the AWS account and deploy your app and you will also get URL of that nextJS deployed application.


d25jem8uvhpvzr.cloudfront.net
Get started by editing app/page.tsx
By Vercel
NEXT.JS
Docs →
Find in-depth information about Next.js features and API.
Learn →
Learn about Next.js in an interactive course with quizzes!
Templates →
Explore the Next.js 13 playground.
Deploy →
Instantly deploy your Next.js site to a shareable URL with Vercel.

7. Check deployed stacks on CloudFormation

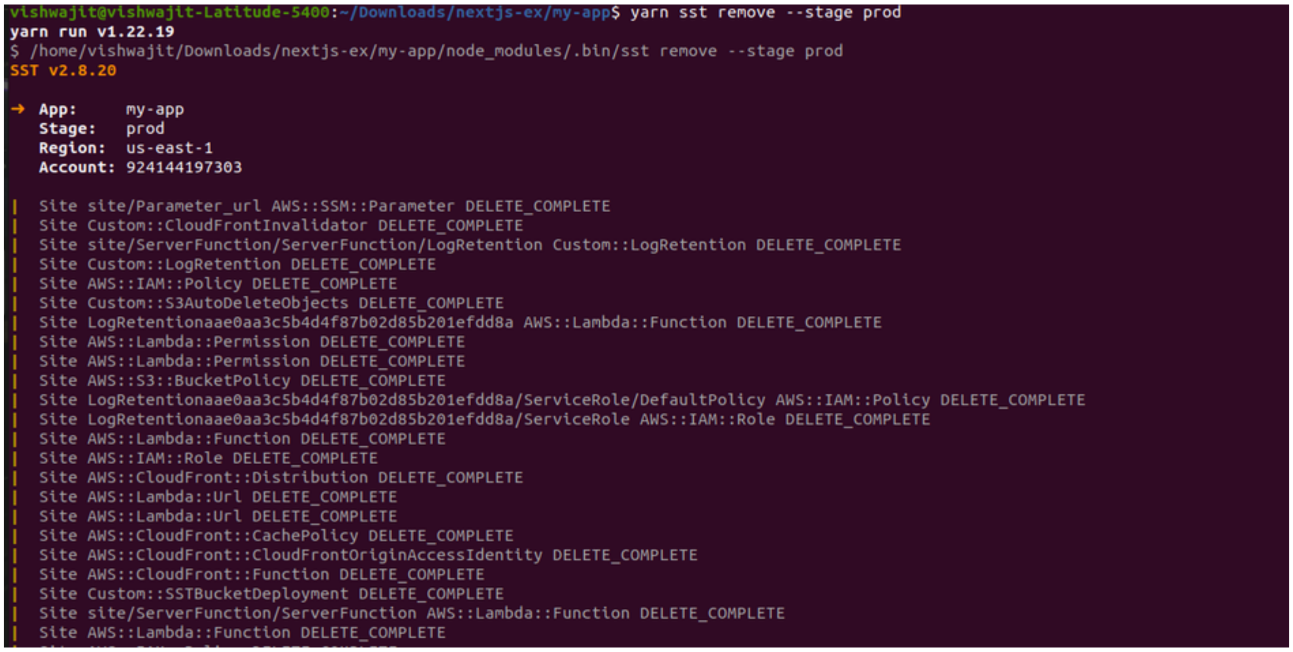


prod-my-app-Site	✓ CREATE_COMPLETE	2023-05-15 13:33:22 UTC+0530	-
SSTBootstrap	✓ CREATE_COMPLETE	2023-05-15 13:29:22 UTC+0530	-
CDKToolkit	✓ CREATE_COMPLETE	2023-05-15 13:28:01 UTC+0530	This stack includes resources needed to deploy AWS CDK apps into this environment

8. Cleaning up

Finally, you can remove the resources created in this example using the following commands.

```
yarn sst remove
yarn sst remove --stage prod
```



```
vishwajit@vishwajit-Latitude-5400:~/Downloads/nextjs-ex/my-app$ yarn sst remove --stage prod
yarn run v1.22.19
$ /home/vishwajit/Downloads/nextjs-ex/my-app/node_modules/.bin/sst remove --stage prod
SST v2.8.20

➜  App:     my-app
   Stage:   prod
   Region:  us-east-1
   Account: 924144197303

|  Site site/Parameter_url AWS::SSM::Parameter DELETE_COMPLETE
|  Site Custom::CloudFrontInvalidator DELETE_COMPLETE
|  Site site/ServerFunction/ServerFunction/LogRetention Custom::LogRetention DELETE_COMPLETE
|  Site Custom::LogRetention DELETE_COMPLETE
|  Site AWS::IAM::Policy DELETE_COMPLETE
|  Site Custom::S3AutoDeleteObjects DELETE_COMPLETE
|  Site LogRetentionaae0aa3c5b4d4f87b02d85b201efdd8a AWS::Lambda::Function DELETE_COMPLETE
|  Site AWS::Lambda::Permission DELETE_COMPLETE
|  Site AWS::Lambda::Permission DELETE_COMPLETE
|  Site AWS::S3::BucketPolicy DELETE_COMPLETE
|  Site LogRetentionaae0aa3c5b4d4f87b02d85b201efdd8a/ServiceRole/DefaultPolicy AWS::IAM::Policy DELETE_COMPLETE
|  Site LogRetentionaae0aa3c5b4d4f87b02d85b201efdd8a/ServiceRole AWS::IAM::Role DELETE_COMPLETE
|  Site AWS::Lambda::Function DELETE_COMPLETE
|  Site AWS::IAM::Role DELETE_COMPLETE
|  Site AWS::CloudFront::Distribution DELETE_COMPLETE
|  Site AWS::Lambda::Url DELETE_COMPLETE
|  Site AWS::Lambda::Url DELETE_COMPLETE
|  Site AWS::CloudFront::CachePolicy DELETE_COMPLETE
|  Site AWS::CloudFront::CloudFrontOriginAccessIdentity DELETE_COMPLETE
|  Site AWS::CloudFront::Function DELETE_COMPLETE
|  Site Custom::SSTBucketDeployment DELETE_COMPLETE
|  Site site/ServerFunction/ServerFunction AWS::Lambda::Function DELETE_COMPLETE
|  Site AWS::Lambda::Function DELETE_COMPLETE
```