
JAVASCRIPT

BACK-END(API WITH DATABASE)

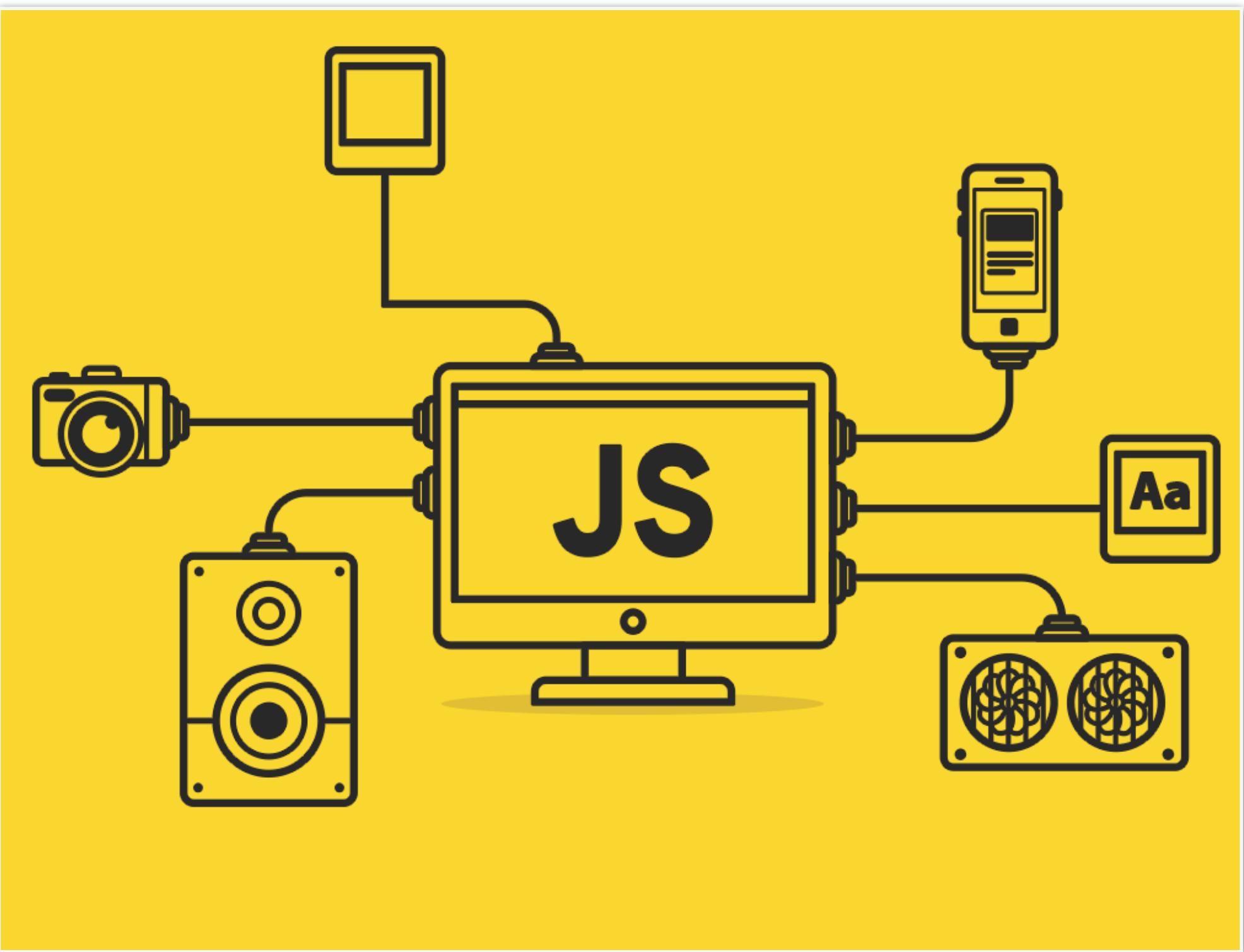
Anirach Mingkhwан

Anirach.m@fitm.kmutnb.ac.th

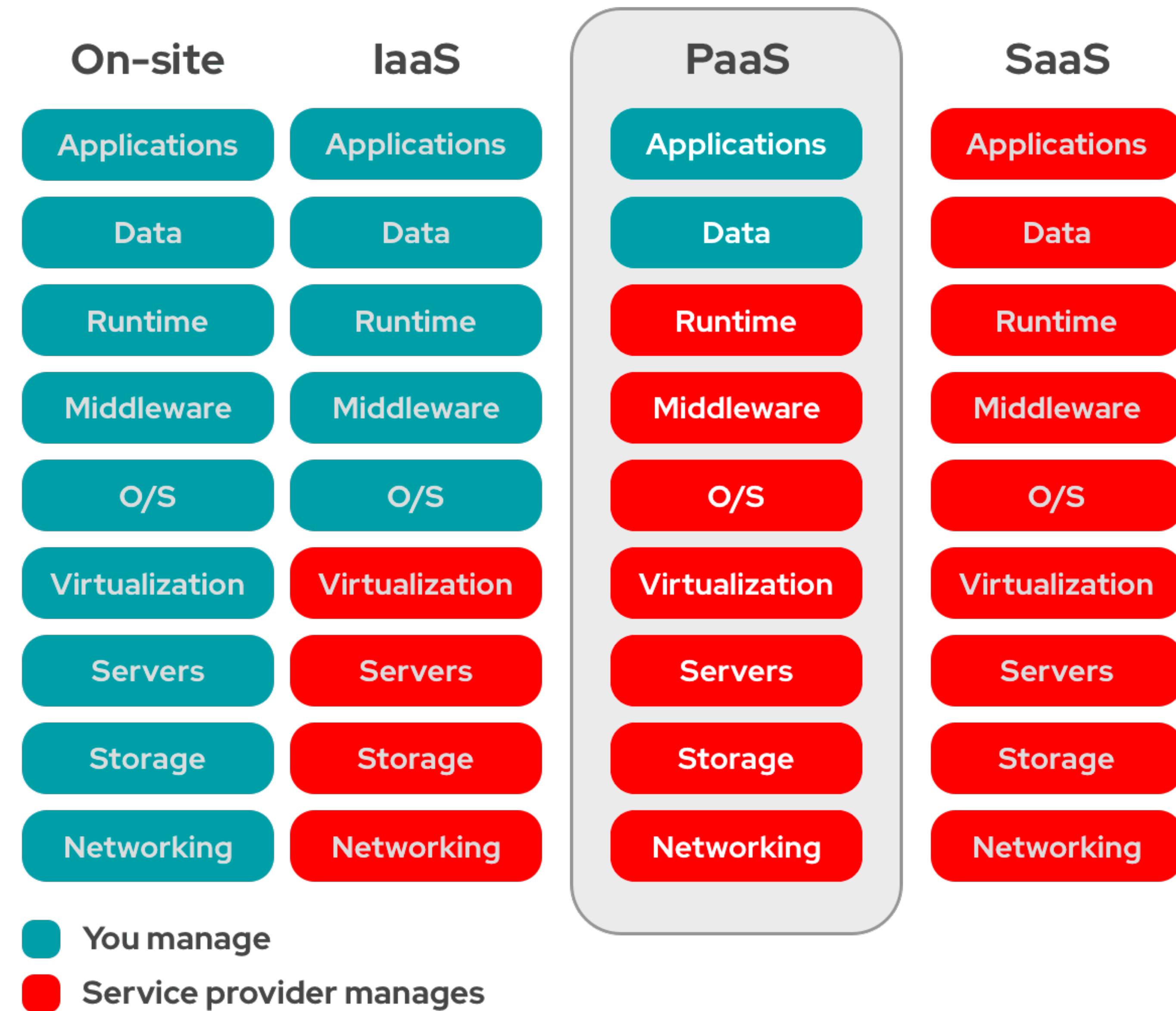


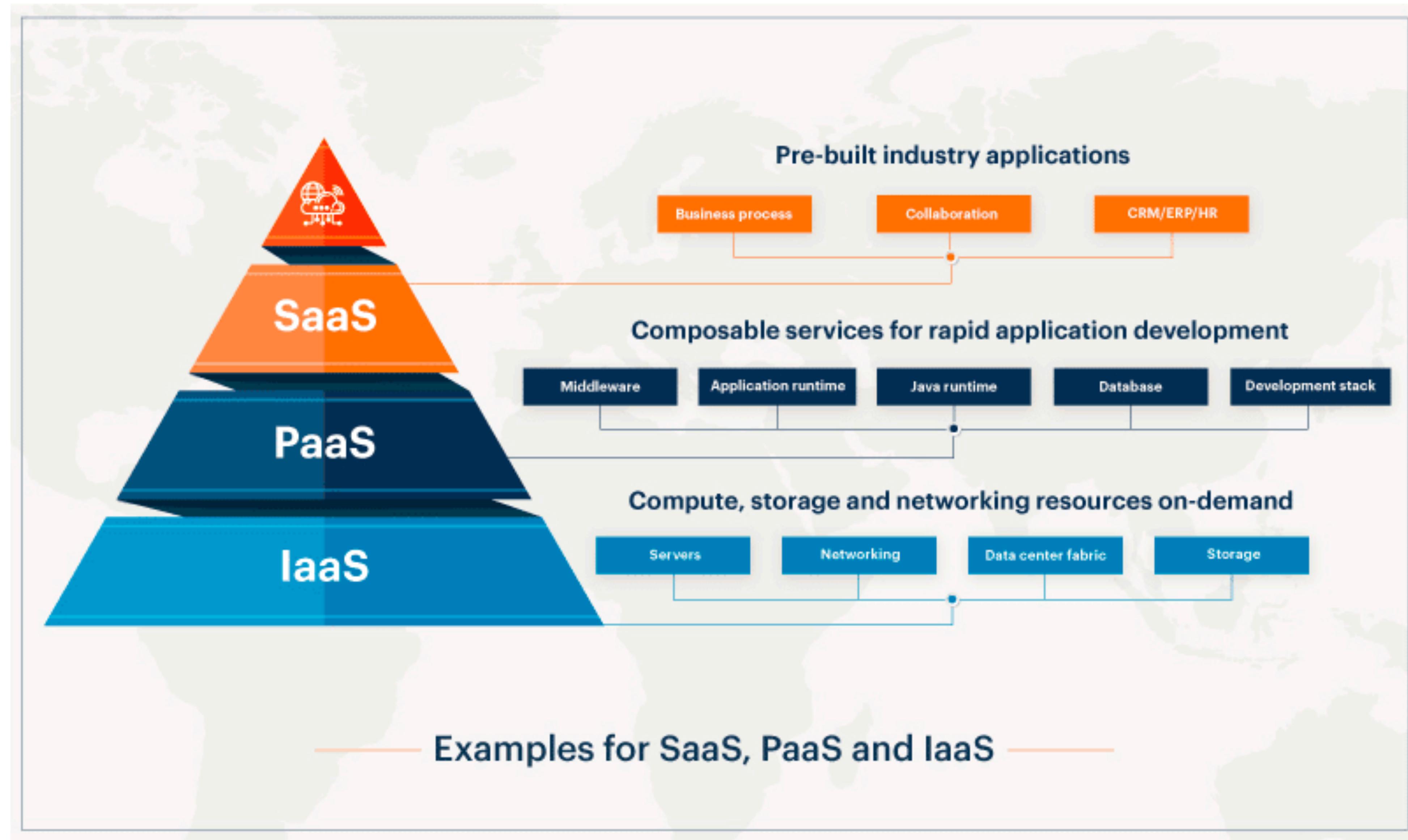
OUTLINE

- PaaS
- Cloud Setting
- Local Database
- SQLite
- SQL command
- API with SQLite DB

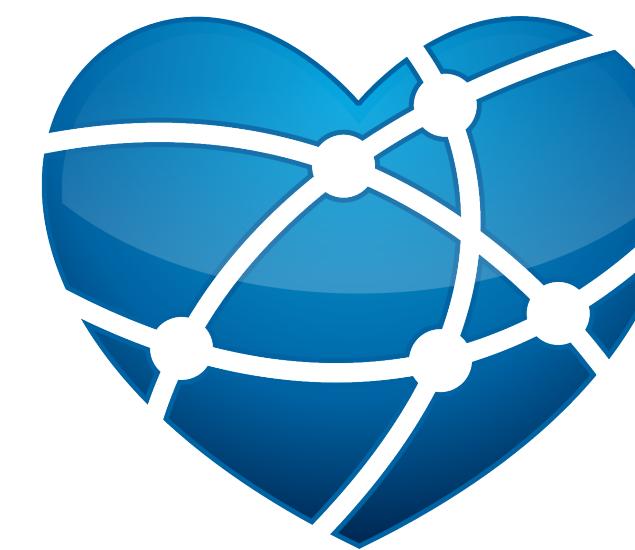


CLOUD PAAS





SETUP CLOUD



RUK-COM
Cloud

<https://app.manage.ruk-com.cloud>

RUK-COM
Cloud บริการของเรา คุณภาพการใช้งาน แจ้งปัญหา แจ้งโอนเงิน สมัครสมาชิก เข้าสู่ระบบ

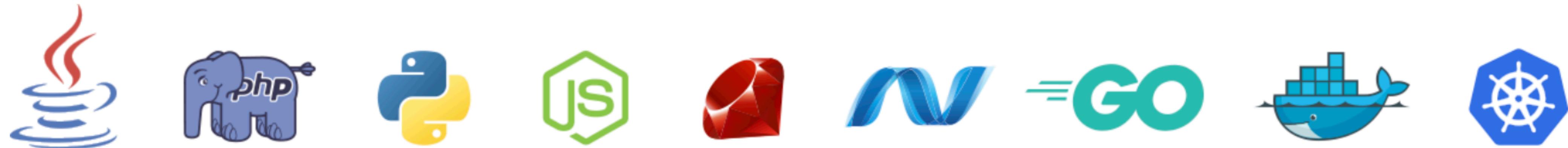
Ruk-Com Cloud

Platform as a service for Developer and Business

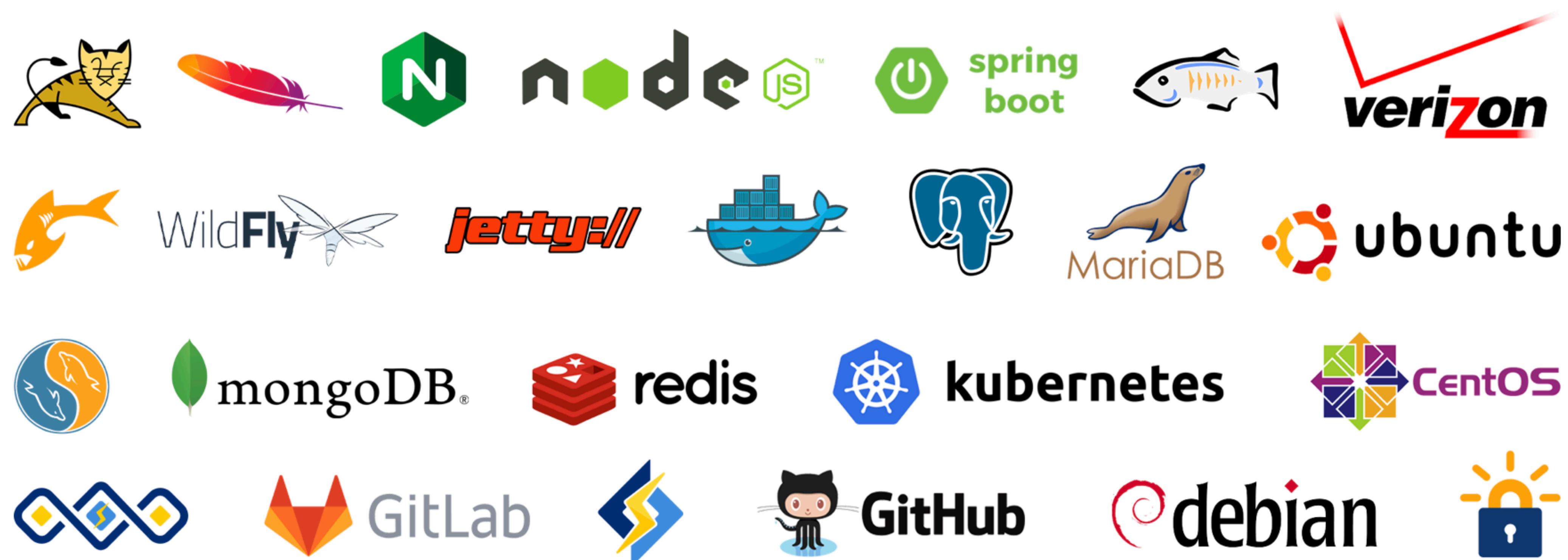
The screenshot shows the Ruk-Com Cloud Platform interface. At the top, there are icons for Java, PHP, Ruby, Node.js, Python, GO Lang, and Docker. Below this is a deployment stack diagram with layers labeled Nginx, MySQL, and MongoDB. To the right, there's a configuration panel for a 'Load Balancer' with 'Vertical Scaling per Node' settings: Reserved 1 cloudlet(s) up to 4 cloudlet(s), and a 'Scaling Limit' up to 512 MiB, 1.6 GHz. It also shows 'Horizontal Scaling' with two nodes. On the far right, there's a summary of resources: 1 cloudlet, Watch later, 100 MHz, and a share icon. It details scaling from 1 to 33 cloudlets, estimated costs from \$1.258 to \$26.068, and a note that unused resources are not charged. Quotas & Pricing and How Pricing Works links are also present.

<https://paas.ruk-com.cloud/>

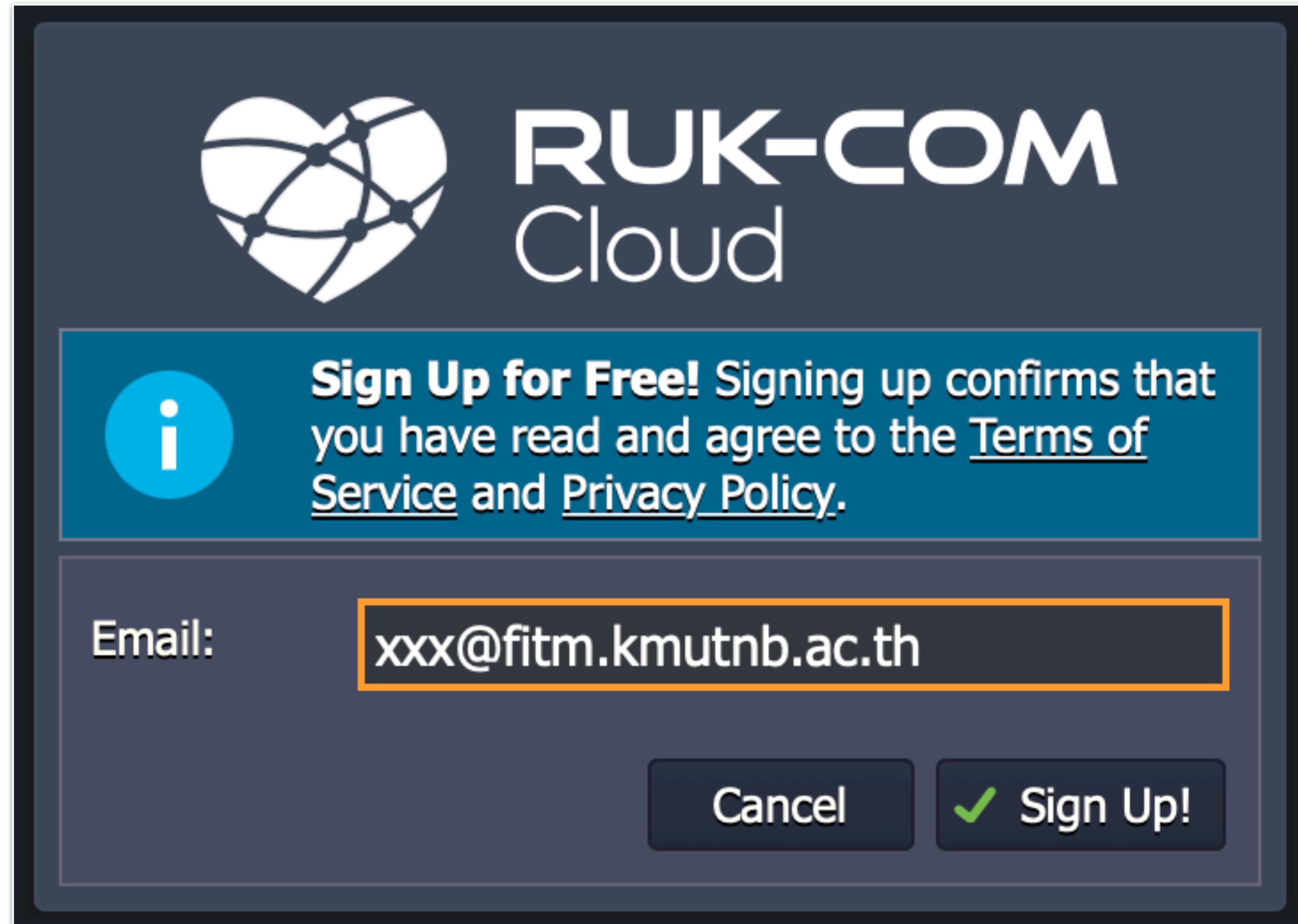
Popular Stacks & Services Out-of-Box Support



New releases and security updates automation, performance and scaling optimization



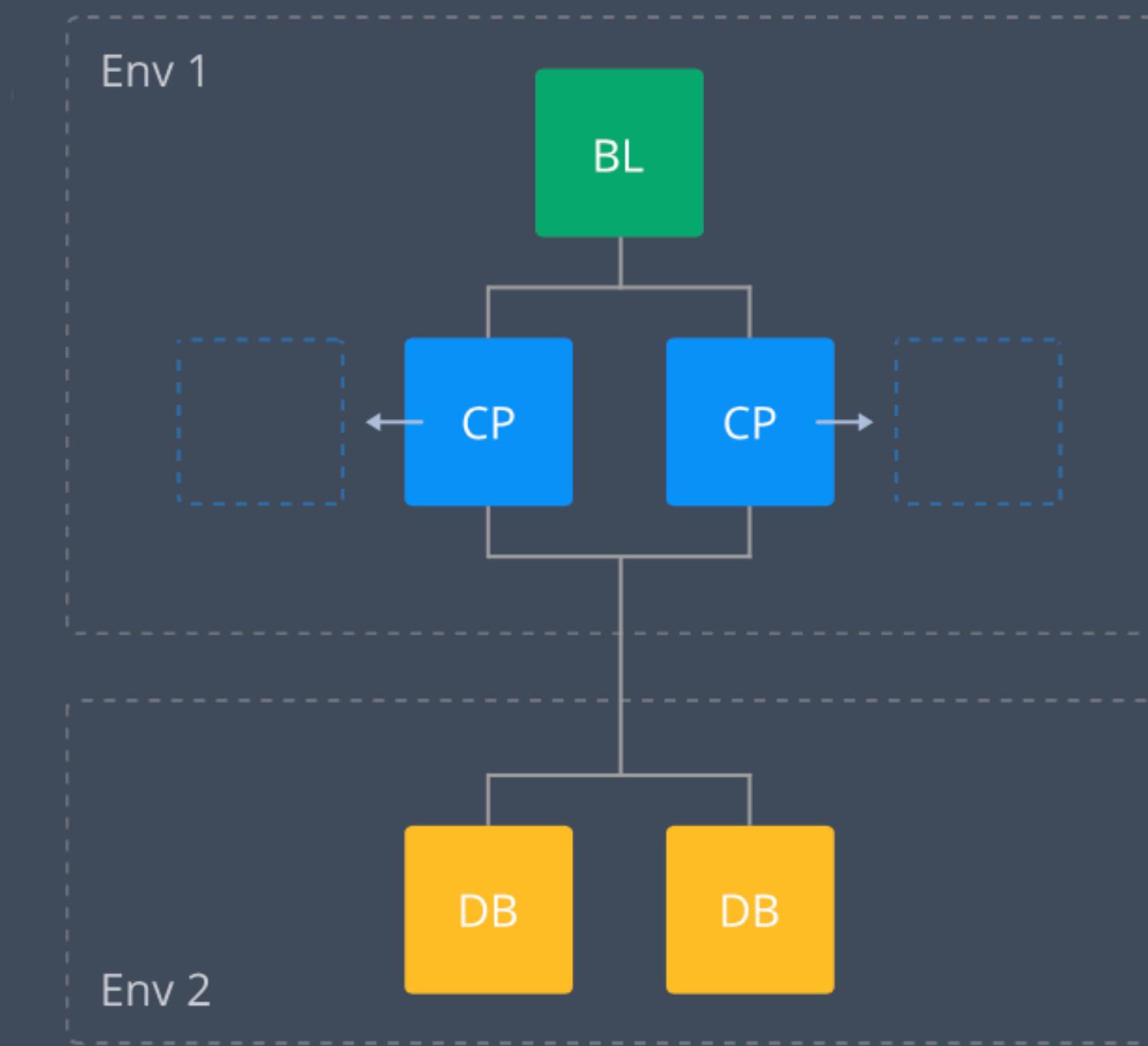
https://app.manage.ruk-com.cloud/?signup=true&group=kmtnb_students



Build Your Application Topology

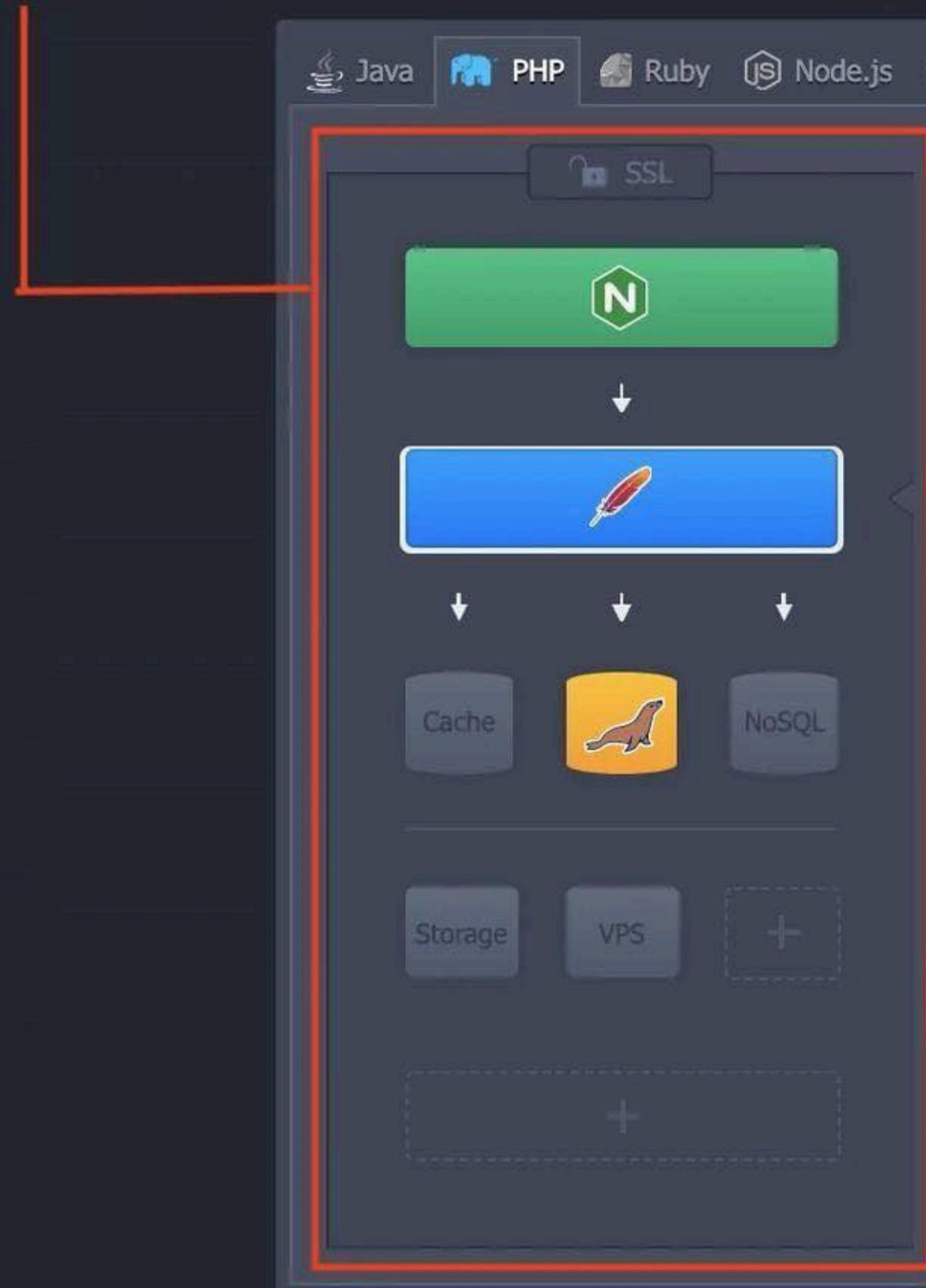


Container
Layer
Environment
Application

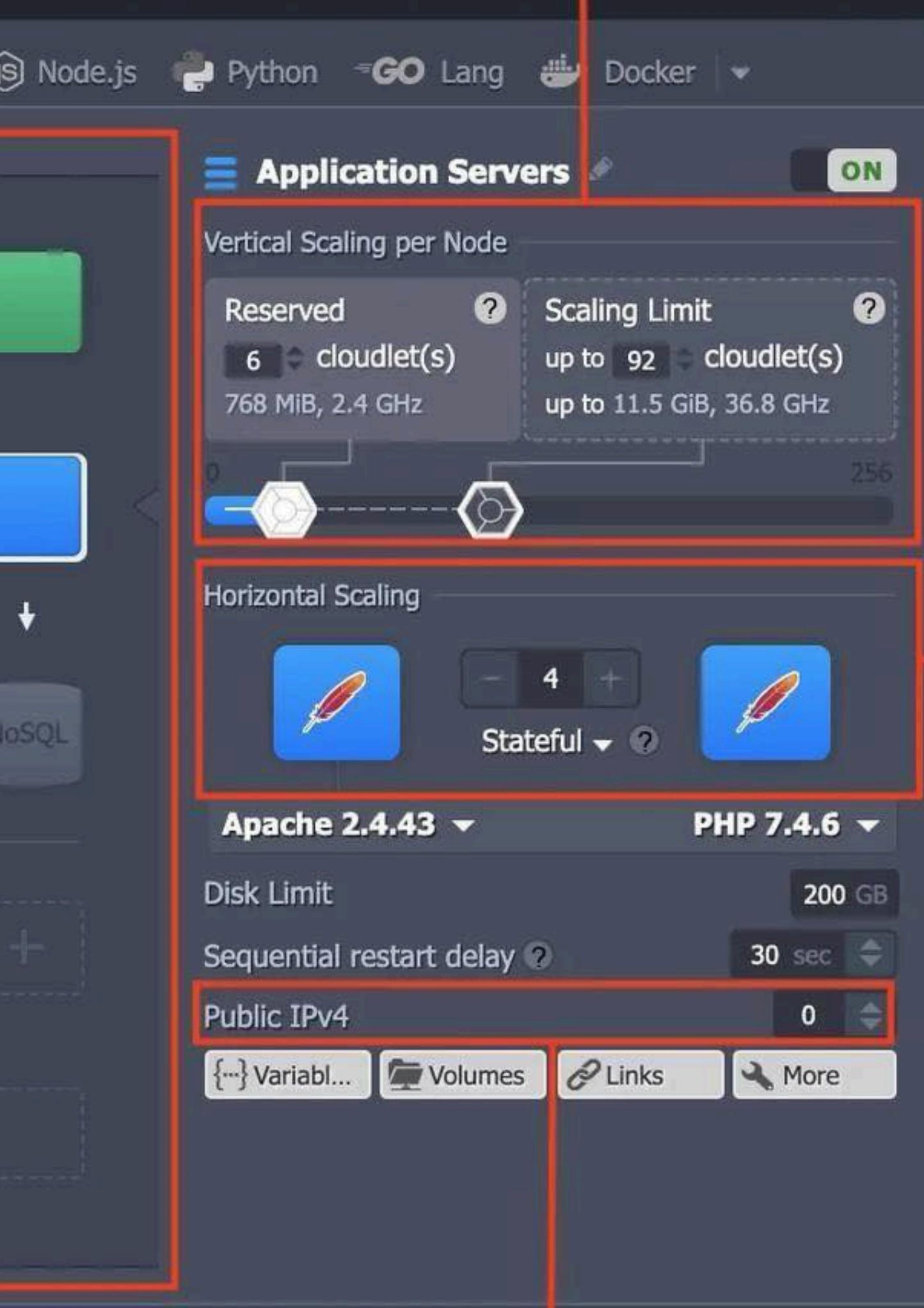


- Balancer (BL)
- Compute (CP)
- Database (DB)
- Storage (ST)

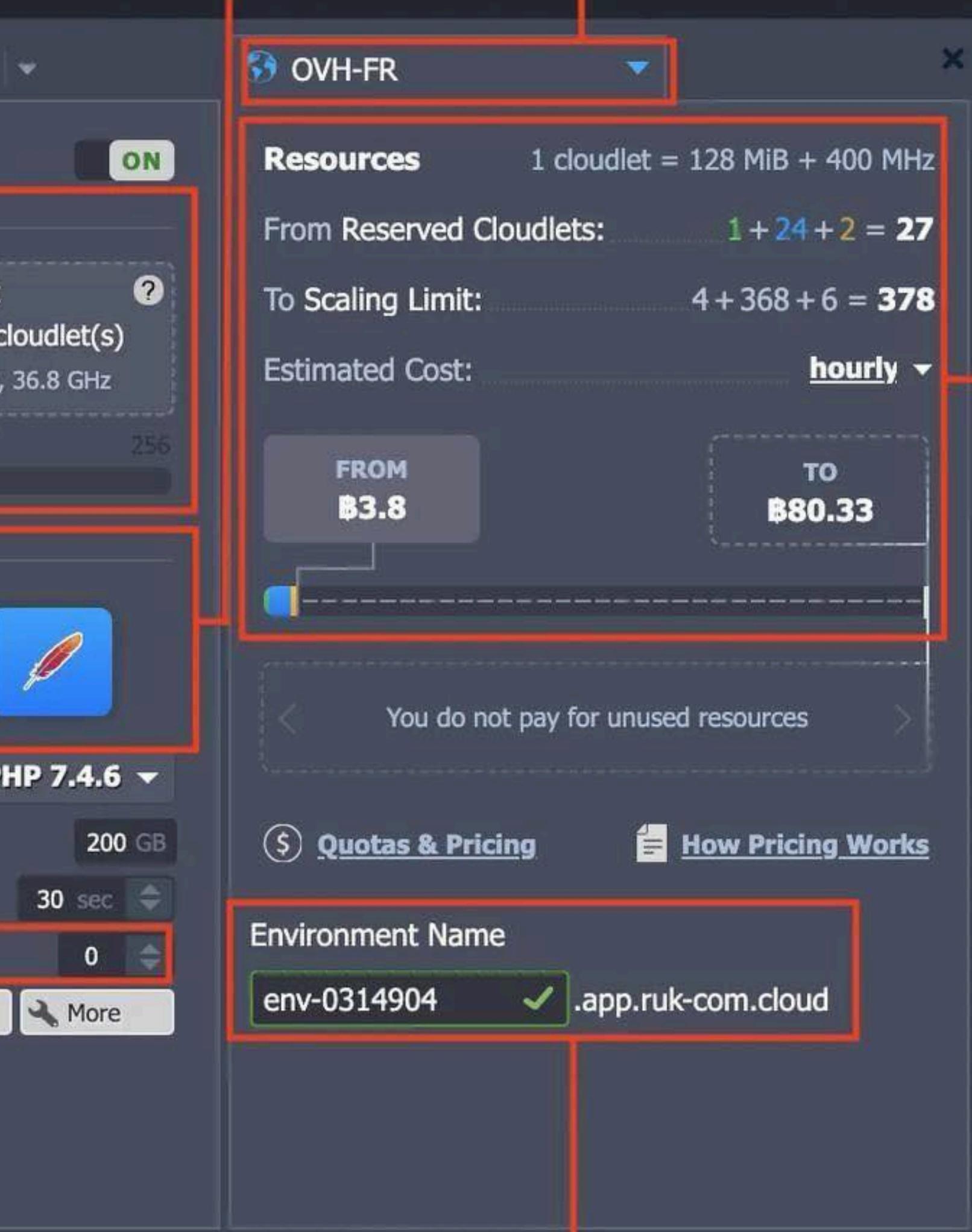
Software Stack



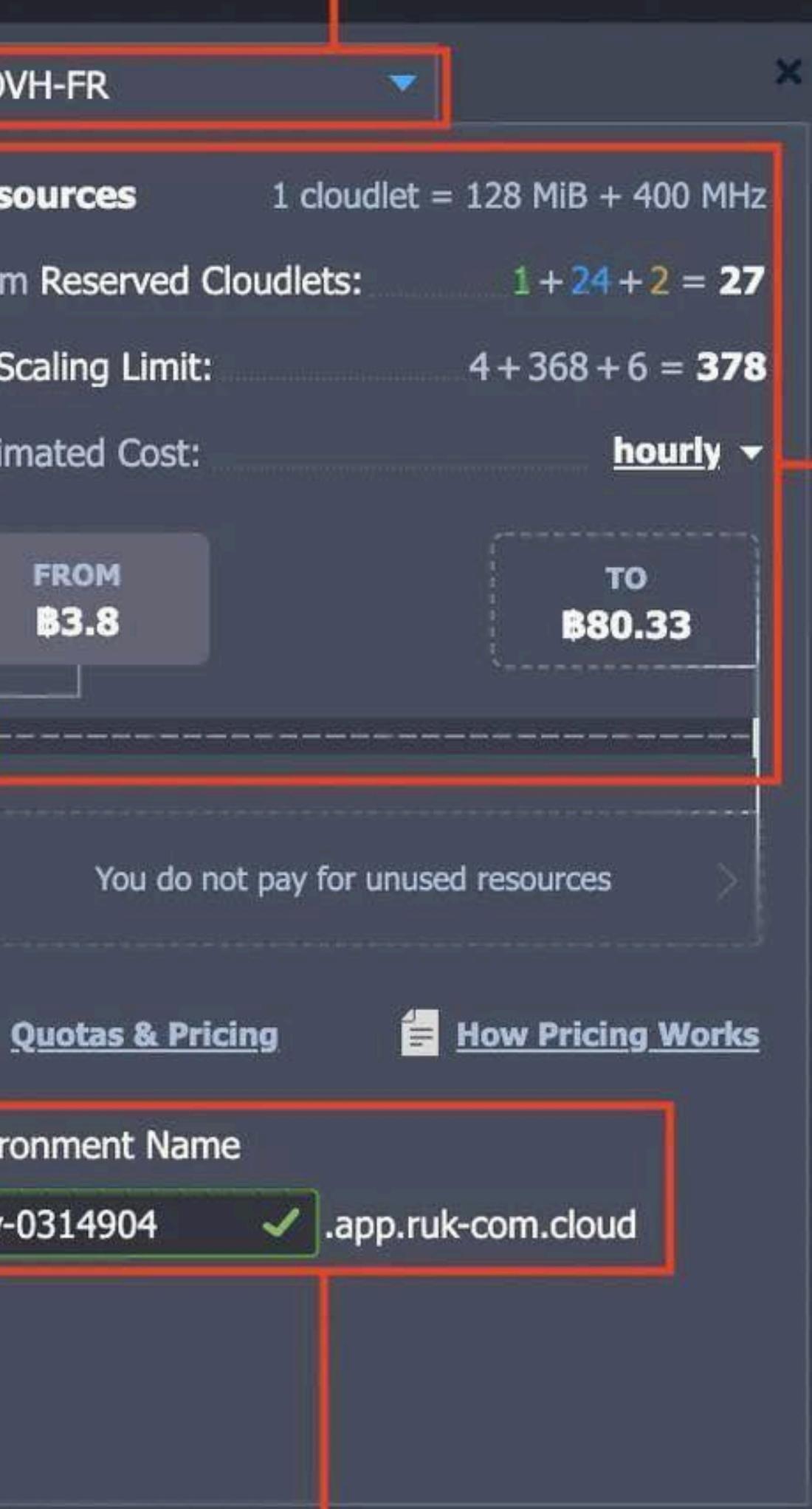
Vertical Scaling



Horizontal scaling



Region



Public IP Address

Environment Name

Resource

The screenshot shows the Rukus Cloud Platform interface for configuring an application server. The main navigation bar includes Java, PHP, Ruby, .NET, Node.js (selected), Python, GO Lang, and Custom.

Application Servers 2 (ON)

Vertical Scaling per Node

- Reserved: 4 cloudlet(s) (512 MiB, 1.6 GHz)
- Scaling Limit: up to 8 cloudlet(s) (up to 1 GiB, 3.2 GHz)

Horizontal Scaling

- Node.js 16.19.... (16.19.0-npm)
- Disk Limit: 100 GB
- Sequential restart delay: 30 sec
- Access via SLB: ON
- Public IPv4: 0

Container Resources

From Reserved Cloudlets: $1 + 8 + 4 = 13$

To Scaling Limit: $4 + 16 + 14 = 34$

Estimated Cost: hourly

FROM **\$0.846** TO **\$2.285**

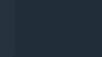
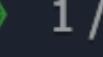
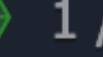
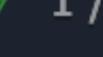
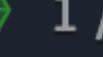
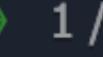
You do not pay for unused resources

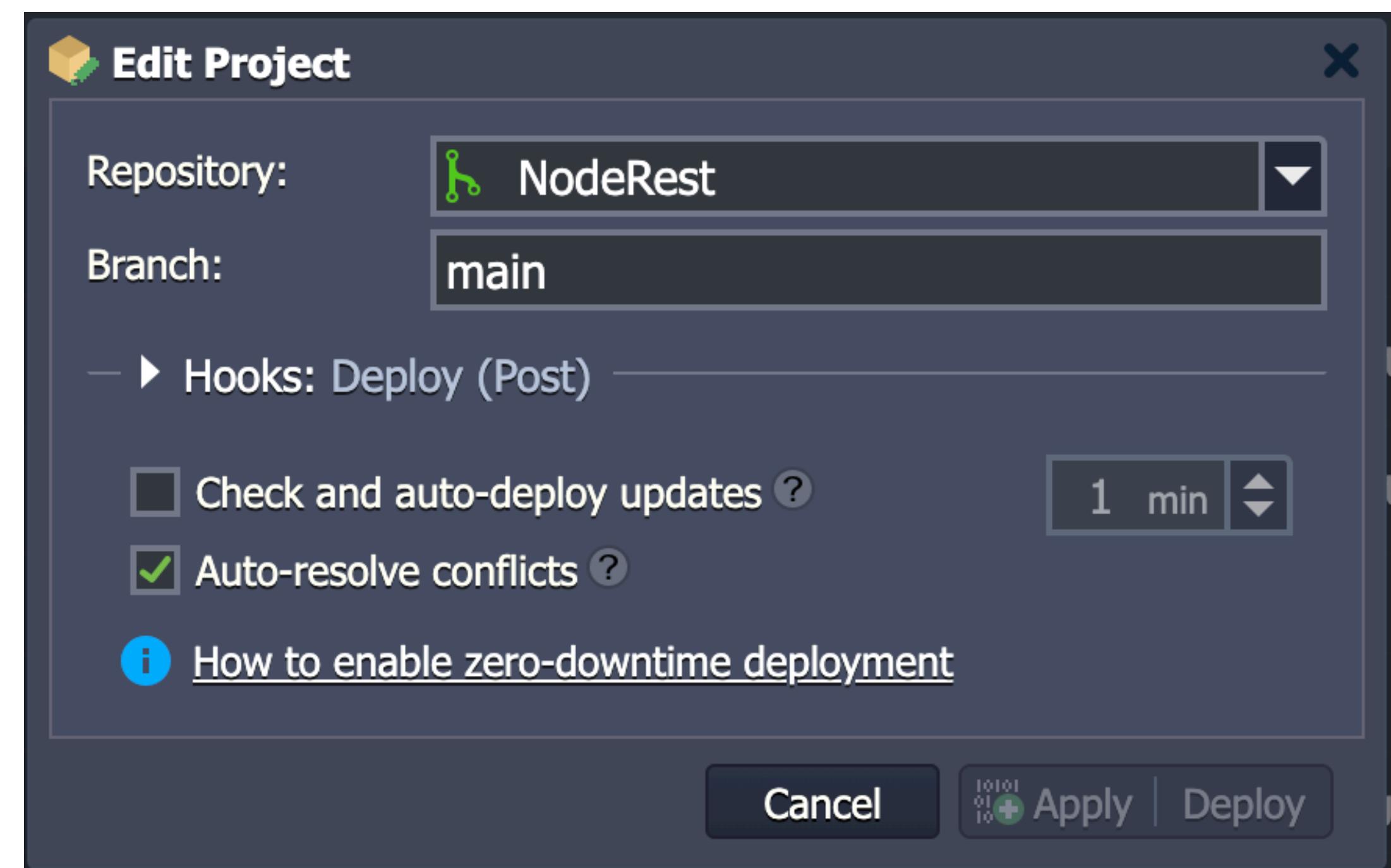
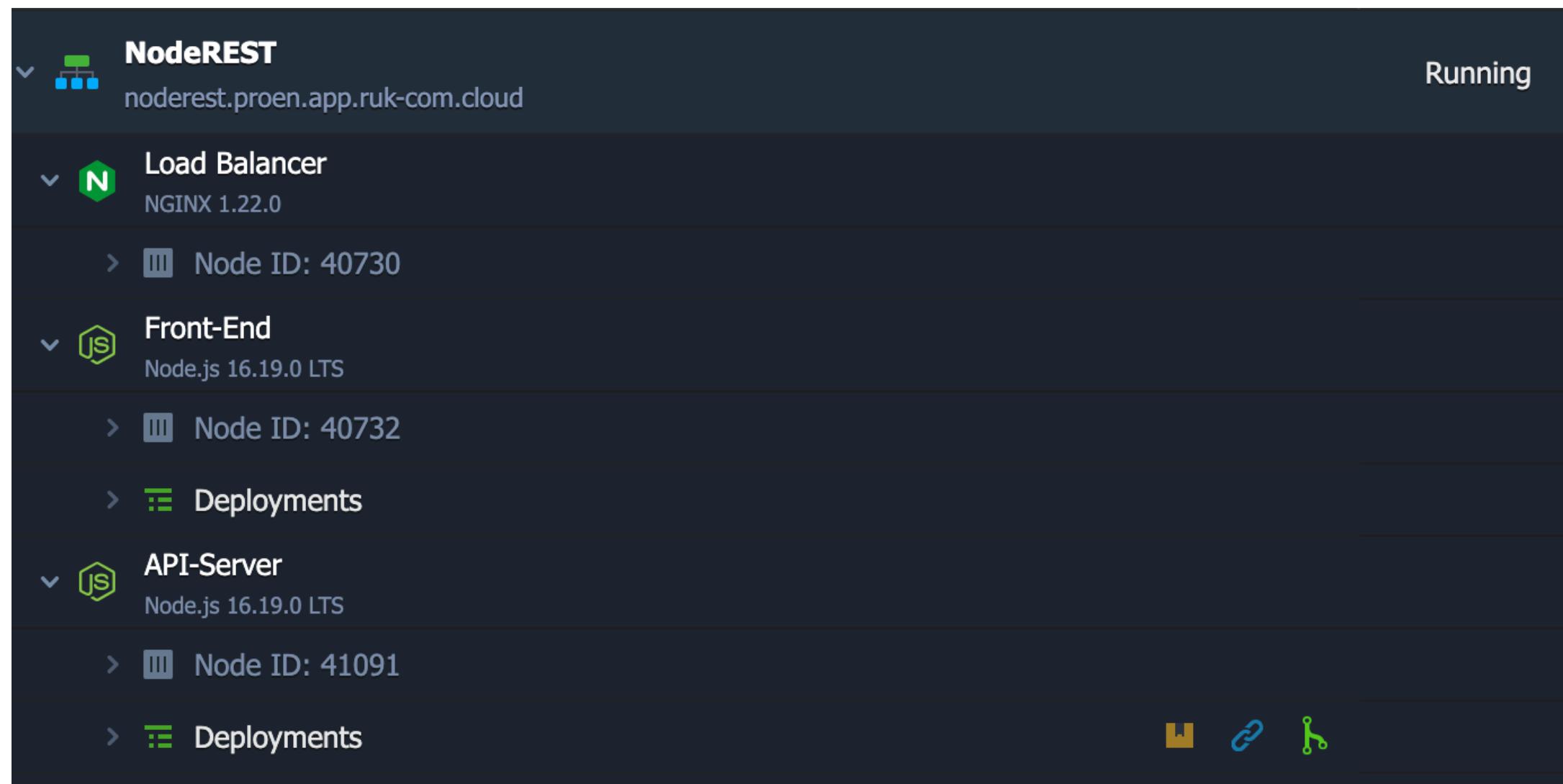
Quotas & Pricing **How Pricing Works**

Environment Name: noderest.proen.app.ruk-com.cloud

Cancel Apply

CONTAINERS	FROM		TO	
	Reserved	Price	Scaling Limit	Price
Load Balancer		\$0.069		\$0.272
	Cloudlets (RAM, CPU)	1	\$0.069	4 \$0.272
Application Servers		\$0.274		\$0.537
	Cloudlets (RAM, CPU)	4	\$0.274	8 \$0.537
SQL Databases		\$0.206		\$0.403
	Cloudlets (RAM, CPU)	3	\$0.206	6 \$0.403
NoSQL Databases		\$0.069		\$0.548
	Cloudlets (RAM, CPU)	1	\$0.069	8 \$0.548
Application Servers 2		\$0.274		\$0.537
	Cloudlets (RAM, CPU)	4	\$0.274	8 \$0.537
Total		\$0.892		\$2.297
Save		\$0.046		\$0.012 ?
Pay Hourly ▾		\$0.846		\$2.285
<u>How to track your spends?</u> Disk Space and Traffic are charged based on consumption.				

		Running		
▼	 NodeREST noderest.proen.app.ruk-com.cloud			 5 / 34
▼	 Load Balancer NGINX 1.22.0			 1 / 4
>	▶  Node ID: 40730	1.22.0		 1 / 4
▼	 Front-End Node.js 16.19.0 LTS		https://github.com/Anirach/Front-End-Node.git	 1 / 8
>	▶  Node ID: 40732	16.19.0-pm2		 1 / 8
>	▶  Deployments			
▼	 API-Server Node.js 16.19.0 LTS		https://github.com/Anirach/NodeREST.git	 1 / 8
>	▶  Node ID: 41091	16.19.0-npm		 1 / 8
>	▶  Deployments			
▼	 SQL Databases PostgreSQL 14.6			 1 / 6
>	▶  Node ID: 40729	14.6		 1 / 6
▼	 NoSQL Databases MongoDB 4.0.2			 1 / 8
>	▶  Node ID: 40731	4.0.2		 1 / 8



[Anirach / NodeREST](#) Private

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)

[main](#) [1 branch](#) [0 tags](#)

 [Anirach](#) Merge branch 'main' of <https://github.com/Anirach/NodeREST>

Database	check SQLite3
NodeExpressREST	update
docs	AddFrontBack Folder
src	Merge branch 'main' of https://github.com/Anirach/NodeREST

[Go to file](#) [Add file](#) [Code](#)

[Local](#) [Codespaces](#)

[Clone](#) [HTTPS](#) [SSH](#) [GitHub CLI](#)

<https://github.com/Anirach/NodeREST.git> [Copy](#)

Use Git or checkout with SVN using the web URL.

About

No description provided.

[Readme](#)

[0 stars](#)

[1 watching](#)

[0 forks](#)

Add Repository

Name:

[Git](#) [SVN](#)

URL:

Branch:

Use Authentication

Login:

Access Type: Token SSH Key

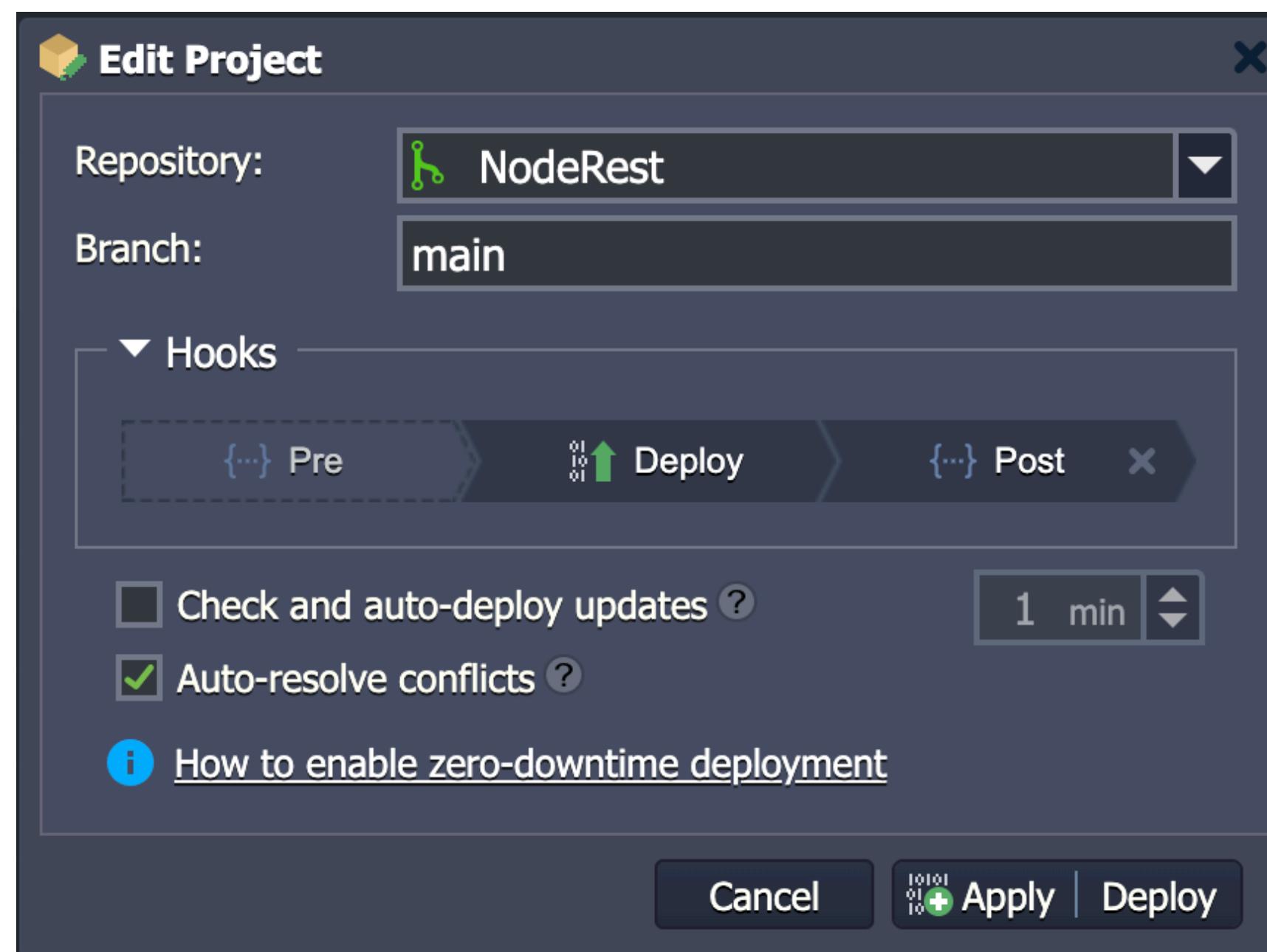
Token:

[Cancel](#) [Add](#)

github.com/settings/tokens

The screenshot shows the GitHub settings page for tokens. On the left, there's a sidebar with options like GitHub Apps, OAuth Apps, Personal access tokens (selected), Fine-grained tokens (Beta), and Tokens (classic). The main area is titled "Personal access tokens (classic)" and shows a table of tokens. One token, "Token to deploy GitHub to RukCom — repo", is listed with a note: "This token has no expiration date." Buttons for "Generate new token" (Beta) and "Revoke all" are at the top right. A modal window titled "Add Repository" is overlaid on the page. It has fields for "Name" (redacted), "URL" (https://... git://... ftp://... example.c), "Branch" (master), and "Use Authentication" checked. Under authentication, it shows "Login" (redacted), "Access Type" (Token selected), and "Token" (redacted). At the bottom are "Cancel" and "Add" buttons.

<https://github.com/settings/tokens>

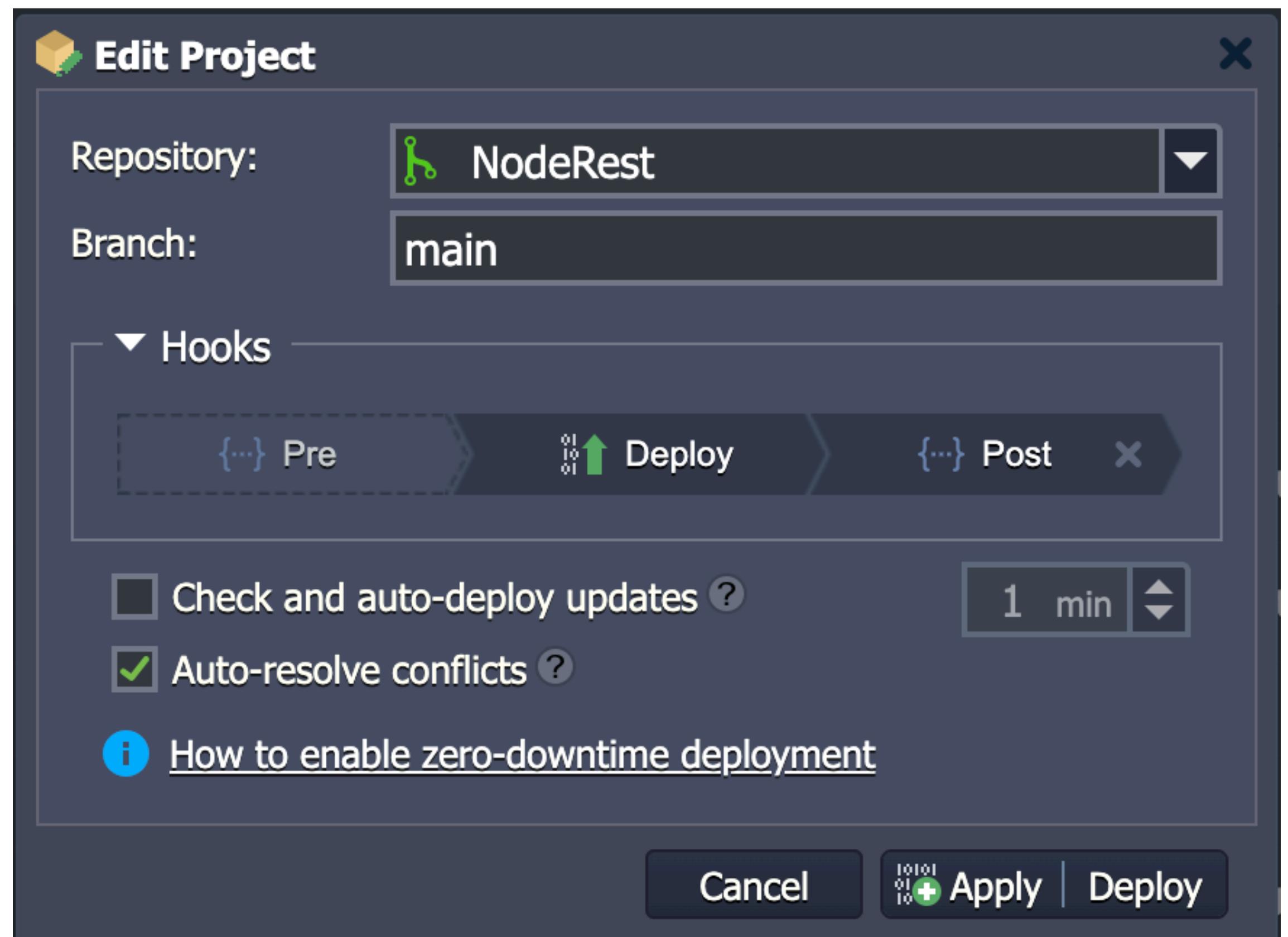


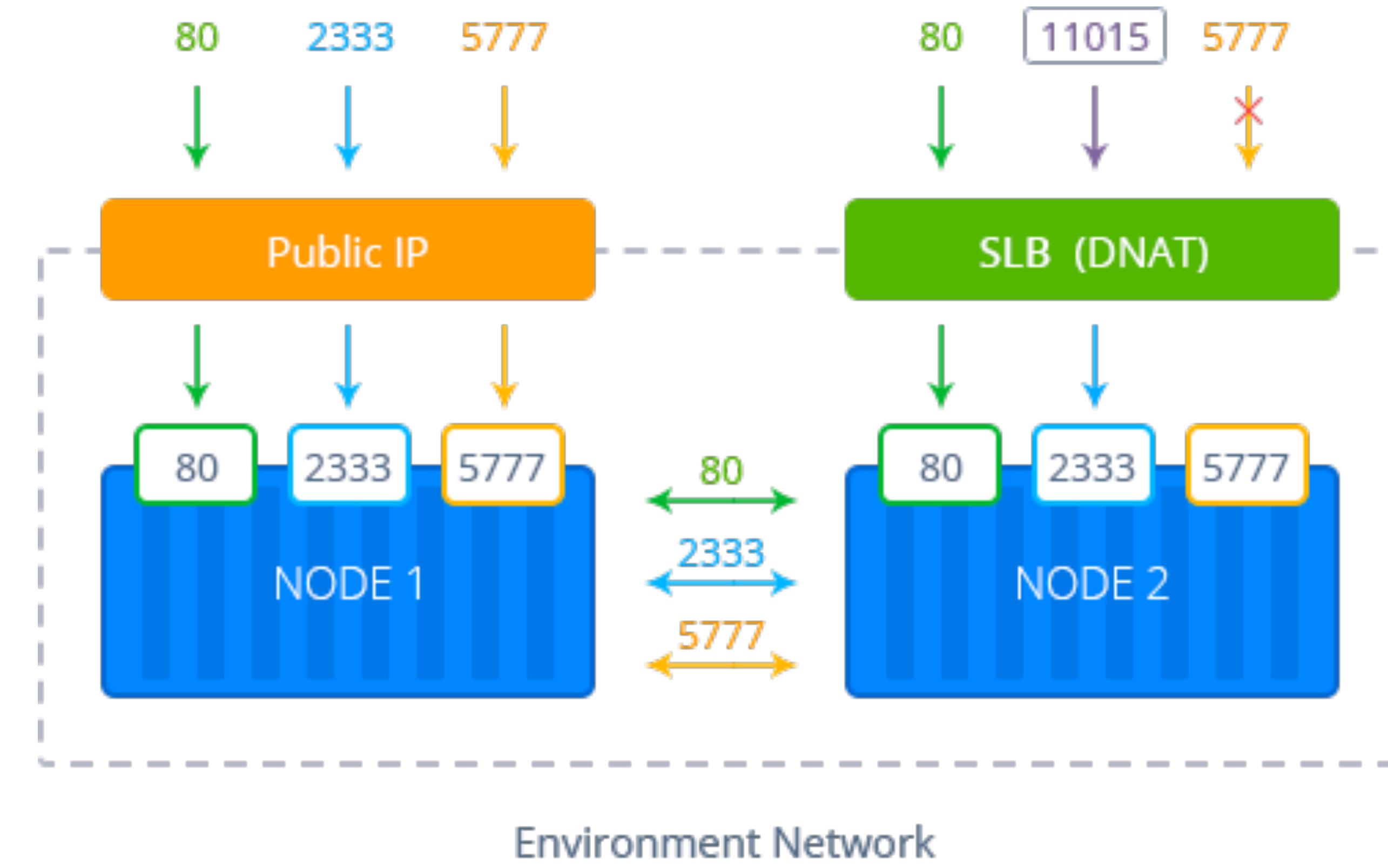
The screenshot shows the 'Edit Post-Deploy Hook' dialog with the following content:

Edit Post-Deploy Hook

```
cd ROOT
killall -9 node
npm install
nohup node src/index.js > runerror.log 2>&1 &
```

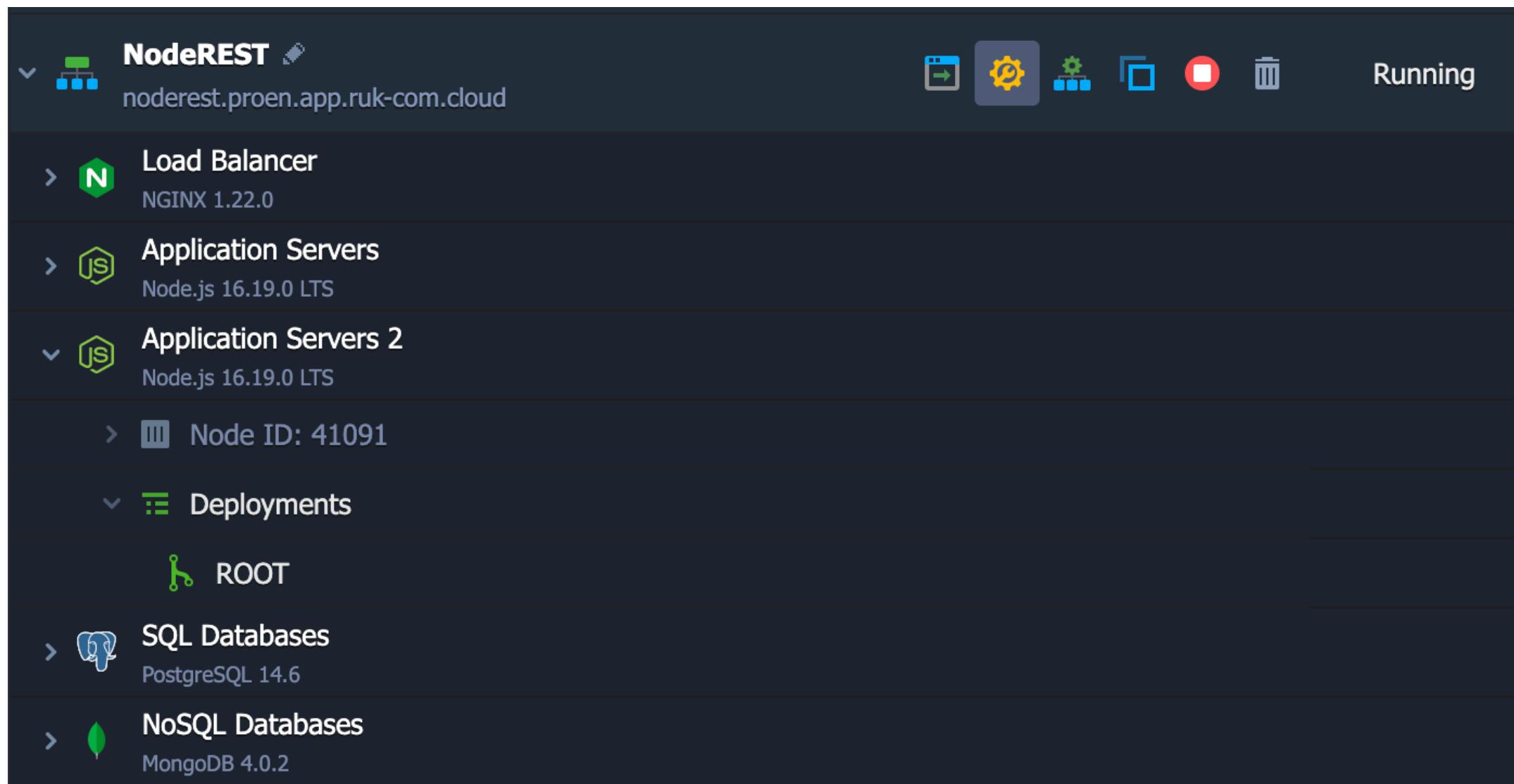
Buttons: Cancel, Apply.





Environment Network

- ↓ Open by default (80, 8080, 8686, 8443, 4848, 4949, 7979)
- ↓ Endpoint - random port provided by platform
- ✗ Dead port - not accessible from outside
- Shared Load Balancer



Settings

Here you can add and manage endpoints, which can be used by other resources for communication

Node	Name	Private Port	Protocol
Application Servers	Node.js 16.19.0 LTS		
Node ID: 40732	WebExpress	3000	TCP
Application Servers 2	Node.js 16.19.0 LTS		
Node ID: 41091	Test	3000	TCP
NoSQL Databases	MongoDB 4.0.2		
Node ID: 40731	MongoDB	27017	TCP

Endpoints

- Add
- Edit
- Remove
- Refresh

The sidebar also includes links for Custom Domains, Custom SSL, SSH Access, Firewall, Load Alerts, Auto Horizontal Scaling, Collaboration, Change Owner, Migration, and Export.

The screenshot shows the Ruk-Com Cloud interface for managing a Node.js application. On the left, the navigation sidebar lists various services: Docalyst, NodeREST, Load Balancer, and Application Servers. The Application Servers section is expanded, showing Node.js 16.19.0 LTS. A modal window titled "Add Endpoint" is open, prompting for configuration details. The "Node:" dropdown is set to "Node ID: 40730". The "Name:" field is empty. The "Private Port:" field contains "3000". The "Protocol:" dropdown is set to "TCP". The "Public Port:" field is set to "Will be set automatically". The "Access URL:" field displays "node40730-noderest.proen.app.ruk-com.cloud:11243". In the background, a list of nodes is visible, with "Node ID: 40730" highlighted in brown.

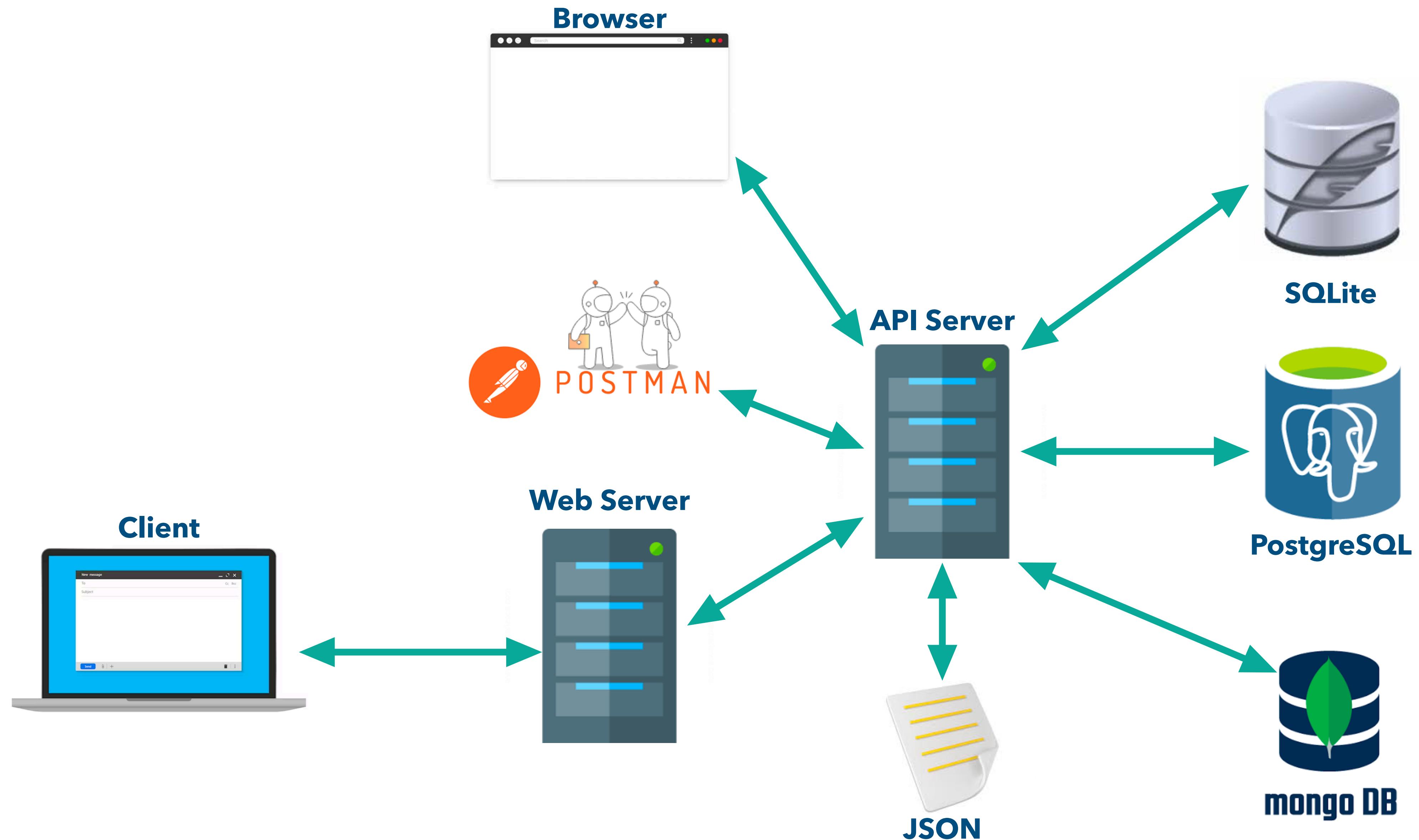
Application Servers 2					
Node.js 16.19.0 LTS					
• Node ID: 41091	Test	3000	TCP	11243	node41091-noderest.proen.app.ruk-com.cloud:11243

← → ⌂

⚠ Not Secure | node41091-noderest.proen.app.ruk-com.cloud:11243

Hello World! I am Happy

LOCAL DATABASE



SQLITE

SQLITE

npmjs.com/package/sqlite3

Natural Polyglot Machine Pro Teams Pricing Documentation

npm Search packages Search Sign Up Sign In

sqlite3 TS

5.1.4 • Public • Published 2 months ago

[Readme](#) [Code](#) Beta [4 Dependencies](#) [3,036 Dependents](#) [99 Versions](#)

 **node-sqlite3**

Asynchronous, non-blocking **SQLite3** bindings for **Node.js**.

release v5.1.4 CI passing license scan passing N-API v3 N-API v6

Features

- Straightforward query and parameter binding interface
- Full Buffer/Blob support
- Extensive **debugging support**
- **Query serialization API**
- **Extension support**, including bundled support for the **json1 extension**
- Big test suite
- Written in modern C++ and tested for memory leaks
- Bundles SQLite v3.40.0, or you can build using a local SQLite

Install

```
> npm i sqlite3
```

Repository

github.com/TryGhost/node-sqlite3

Homepage

github.com/TryGhost/node-sqlite3

Weekly Downloads

704,764



Version

5.1.4

License

BSD-3-Clause

npm i sqlite3

SQLITE VIEWER

Extension: SQLite Viewer X

 **SQLite Viewer** v0.1.5

Florian Klampfer | 307,191 | ★★★★★(23)

SQLite Viewer for VSCode

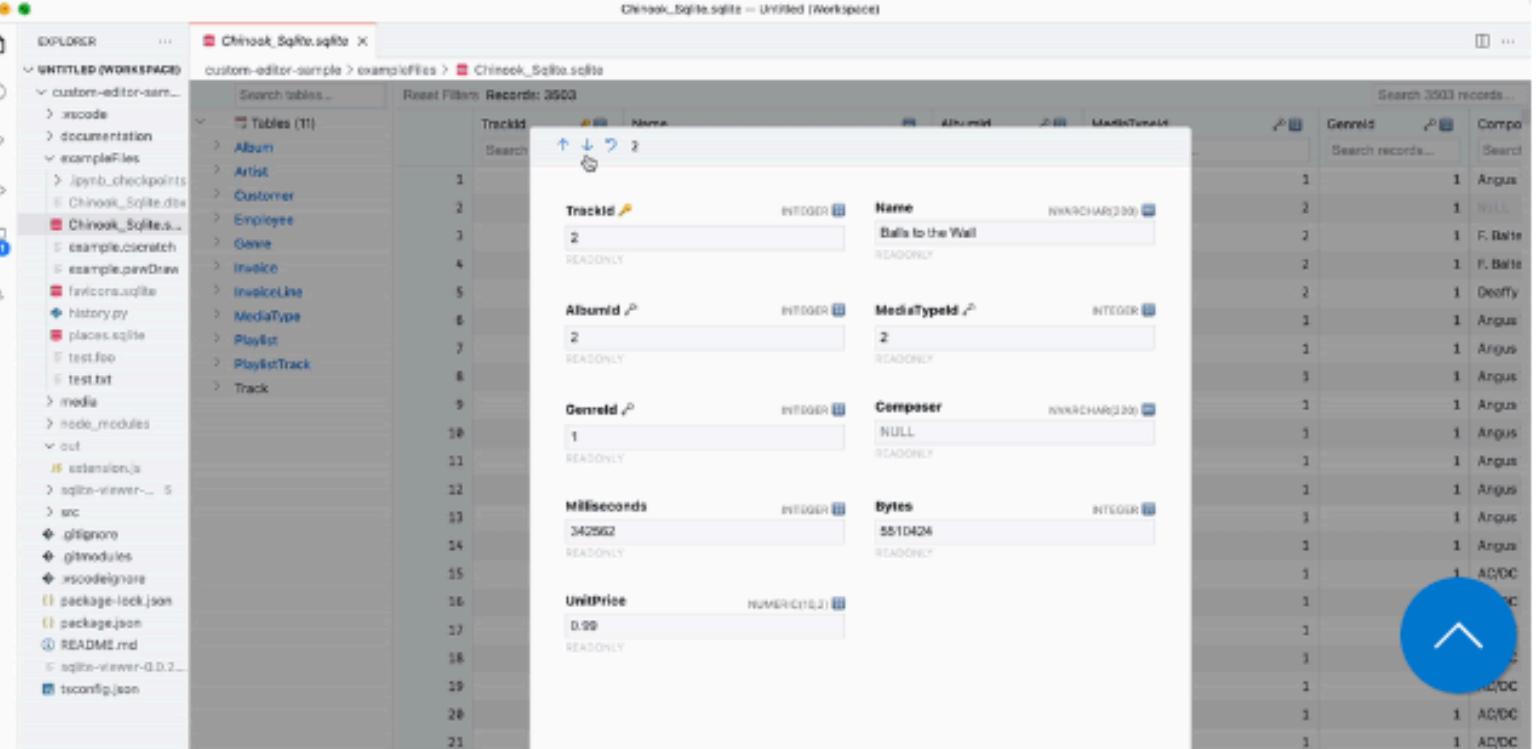
[Disable](#) | [Uninstall](#) | [Switch to Pre-Release Version](#)

This extension is enabled globally.

[DETAILS](#) [FEATURE CONTRIBUTIONS](#) [CHANGELOG](#) [RUNTIME STATUS](#)

SQLite Viewer for VSCode

A quick and easy SQLite viewer for VSCode, inspired by DB Browser for SQLite and Airtable.



Categories

- Other

Extension Resources

[Marketplace](#) [Repository](#) [License](#) [Florian Klampfer](#)

More Info

Published	10/1/2021, 15:04:30
Last released	3/21/2022, 11:57:39
Last updated	10/1/2022, 17:34:16
Identifier	qwtel.sqlite-

The screenshot shows a web browser window with the URL "sqlitetutorial.net" in the address bar. The page header features the "SQLITE TUTORIAL" logo with a stylized green 'S' icon. A navigation menu at the top includes links for "HOME", "START HERE", "VIEWS", "INDEXES", and "TRIGGERS". Below the menu, the title "SQLite Tutorial" is displayed in a large, bold, orange font.

SQLite Tutorial

This **SQLite tutorial** teaches you everything you need to know to start using SQLite effectively. In this tutorial, you will learn SQLite step by step through extensive hands-on practices.

This SQLite tutorial is designed for developers who want to use SQLite as the back-end database or to use SQLite to manage structured data in applications including desktop, web, and mobile apps.

SQLite is an open-source, zero-configuration, self-contained, stand-alone, transaction relational database engine designed to be embedded into an application.



Getting started with SQLite

You should go through this section if this is the first time you have worked with SQLite. Follow these 4-easy steps to get started with SQLite fast.

- First, help you answer the first and important question: [what is SQLite?](#) You will have a brief overview of SQLite.
- Second, show you step-by-step how to download and install the [SQLite tools](#) on your computer.

<https://www.sqlitetutorial.net/>

SQL COMMAND

```
// create books table if it doesn't exist
db.run(`CREATE TABLE IF NOT EXISTS books (
    id INTEGER PRIMARY KEY,
    title TEXT,
    author TEXT
)`);
```

The screenshot shows the homepage of SQLBolt.com. At the top, there's a navigation bar with a lock icon and 'sqlbolt.com' in the address bar, and various browser icons on the right. Below the header, the SQLBolt logo (a blue cylinder with a lightning bolt) and the text 'Learn SQL with simple, interactive exercises.' are displayed. To the right are links for 'Interactive Tutorial' (with a graduation cap icon) and 'More Topics' (with a square icon). The main content area has a blue header bar with the title 'Introduction to SQL'. Below it, a paragraph welcomes users to SQLBolt, stating it's a series of interactive lessons and exercises designed to help learn SQL right in the browser. A section titled 'What is SQL?' follows, explaining that SQL is a language for querying, manipulating, and transforming data from relational databases. It highlights its simplicity and widespread use for safe and scalable storage. A 'Did you know?' box provides information about the popularity of SQL databases like SQLite, MySQL, PostgreSQL, Oracle, and Microsoft SQL Server, noting that while they support common SQL standards, they have different features and storage types. The bottom section, titled 'Relational databases', discusses what they are and provides an example of a vehicle database.

SQLBolt

Learn SQL with simple, interactive exercises.

Interactive Tutorial More Topics

Introduction to SQL

Welcome to SQLBolt, a series of interactive lessons and exercises designed to help you quickly learn SQL right in your browser.

What is SQL?

SQL, or Structured Query Language, is a language designed to allow both technical and non-technical users query, manipulate, and transform data from a relational database. And due to its simplicity, SQL databases provide safe and scalable storage for millions of websites and mobile applications.

Did you know?

There are many popular SQL databases including SQLite, MySQL, Postgres, Oracle and Microsoft SQL Server. All of them support the common SQL language standard, which is what this site will be teaching, but each implementation can differ in the additional features and storage types it supports.

Relational databases

Before learning the SQL syntax, it's important to have a model for what a relational database actually is. A relational database represents a collection of related (two-dimensional) tables. Each of the tables are similar to an Excel spreadsheet, with a fixed number of named columns (the attributes or properties of the table) and any number of rows of data.

For example, if the Department of Motor Vehicles had a database, you might find a table containing all the known vehicles that people in the state are driving. This table might need to store the model name, type, number of wheels, and number of doors of each vehicle for example.

API-SQL-SQLITE

```
JS CRUDBookSQLite.js U X  
src > JS CRUDBookSQLite.js > ...  
1 // SQLite3 CRUD operations  
2 // npm install sqlite3  
3 // Create a Book.sqlite file in Database folder  
4 // Run this file with node CRUDBookSQLite.js  
5 // Test with Postman  
6  
7 const express = require('express');  
8 const sqlite3 = require('sqlite3');  
9 const app = express();  
10  
11 // connect to database  
12 const db = new sqlite3.Database('./Database/Book.sqlite');  
13  
14 // parse incoming requests  
15 app.use(express.json());  
16  
17 // create books table if it doesn't exist  
18 db.run(`CREATE TABLE IF NOT EXISTS books (  
19     id INTEGER PRIMARY KEY,  
20     title TEXT,  
21     author TEXT  
22 )`);  
23
```

```
JS CRUDBookSQLite.js X
src > JS CRUDBookSQLite.js > ...
22
23
24 // route to get all books
25 app.get('/books', (req, res) => {
26   db.all('SELECT * FROM books', (err, rows) => {
27     if (err) {
28       res.status(500).send(err);
29     } else {
30       res.json(rows);
31     }
32   });
33 });
34
35 // route to get a book by id
36 app.get('/books/:id', (req, res) => {
37   db.get('SELECT * FROM books WHERE id = ?', req.params.id, (err, row) => {
38     if (err) {
39       res.status(500).send(err);
40     } else {
41       if (!row) {
42         res.status(404).send('Book not found');
43       } else {
44         res.json(row);
45       }
46     }
47   });
48 });
49
```

The screenshot shows a code editor window with the title bar "CRUDBookSQLite.js" and a breadcrumb navigation bar "src > CRUDBookSQLite.js > ...". The main area displays a block of JavaScript code for a Node.js application. The code defines two routes: one for creating a new book and another for updating an existing book.

```
49
50 // route to create a new book
51 app.post('/books', (req, res) => {
52   const book = req.body;
53   db.run('INSERT INTO books (title, author) VALUES (?, ?)', book.title, book.author, function(err) {
54     if (err) {
55       res.status(500).send(err);
56     } else {
57       book.id = this.lastID;
58       res.send(book);
59     }
60   });
61 });
62
63 // route to update a book
64 app.put('/books/:id', (req, res) => {
65   const book = req.body;
66   db.run('UPDATE books SET title = ?, author = ? WHERE id = ?', book.title, book.author, req.params.id, function(err) {
67     if (err) {
68       res.status(500).send(err);
69     } else {
70       res.send(book);
71     }
72   });
73 });
74
```

```
JS CRUDBookSQLite.js U X
src > JS CRUDBookSQLite.js > ...
74
75 // route to delete a book
76 app.delete('/books/:id', (req, res) => {
77   db.run('DELETE FROM books WHERE id = ?', req.params.id, function(err) {
78     if (err) {
79       res.status(500).send(err);
80     } else {
81       res.send({});
82     }
83   });
84 });
85
86 const port = process.env.PORT || 3000;
87 app.listen(port, () => console.log(`Listening on port ${port}...`));
88
```

TEST WITH POSTMAN

Home Workspaces API Network Explore Search Postman Invite Gear Bell Upgrade

My Workspace

New Import

GET All Books GET Book By ID POST Create new PUT Update Boo DEL Delete Book + ... No Environment

Books DB Node API / All Books

GET localhost:3000/books Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results Security

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "id": 1,
4     "title": "Fire places",
5     "author": "Goya"
6   },
7   {
8     "id": 3,
9     "title": "7000 Stars",
10    "author": "Anirach"
11  }
12 ]

```

EVENT TIME

EVENT	TIME
Prepare	2.46 ms
Socket Initialization	0.42 ms
DNS Lookup	Cache
TCP Handshake	Cache
Transfer Start	1.93 ms
Download	0.89 ms
Process	0.08 ms
Total	5.77 ms

Online Find and Replace Console Cookies Capture requests Runner Trash

THANK YOU

- PaaS
- Cloud Setting
- Local Database
- SQLite
- SQL command
- API with SQLite DB
- Postman Test