

STREAM SERIES

desvendando a ferramenta

NodeJS e ElasticSearch
Persistindo mensagens do console.log

SERGIO CABRAL



sergiocabral.com

Índice

1. ElasticSearch: Configurando ambiente.....	2
1.1. Criando o servidor	2
1.2. Configurando o servidor	5
1.3. Instalando Elasticsearch e Kibana via Docker.....	5
1.4. Acessando as aplicações	9
1.4.1. Elasticsearch.....	9
1.4.2. Kibana	10
2. NodeJS: Criando uma aplicação base	11
2.1. Criando uma aplicação tipo NPM.....	11
2.2. Se comunicando com o Elasticsearch	12

Este conteúdo é gratuito. Foi preparado como esboço para uma aula do Experts Club, da Rocketseat.

Se tudo estiver certo, você consegue a versão atualizada desse *e-book* nos links:

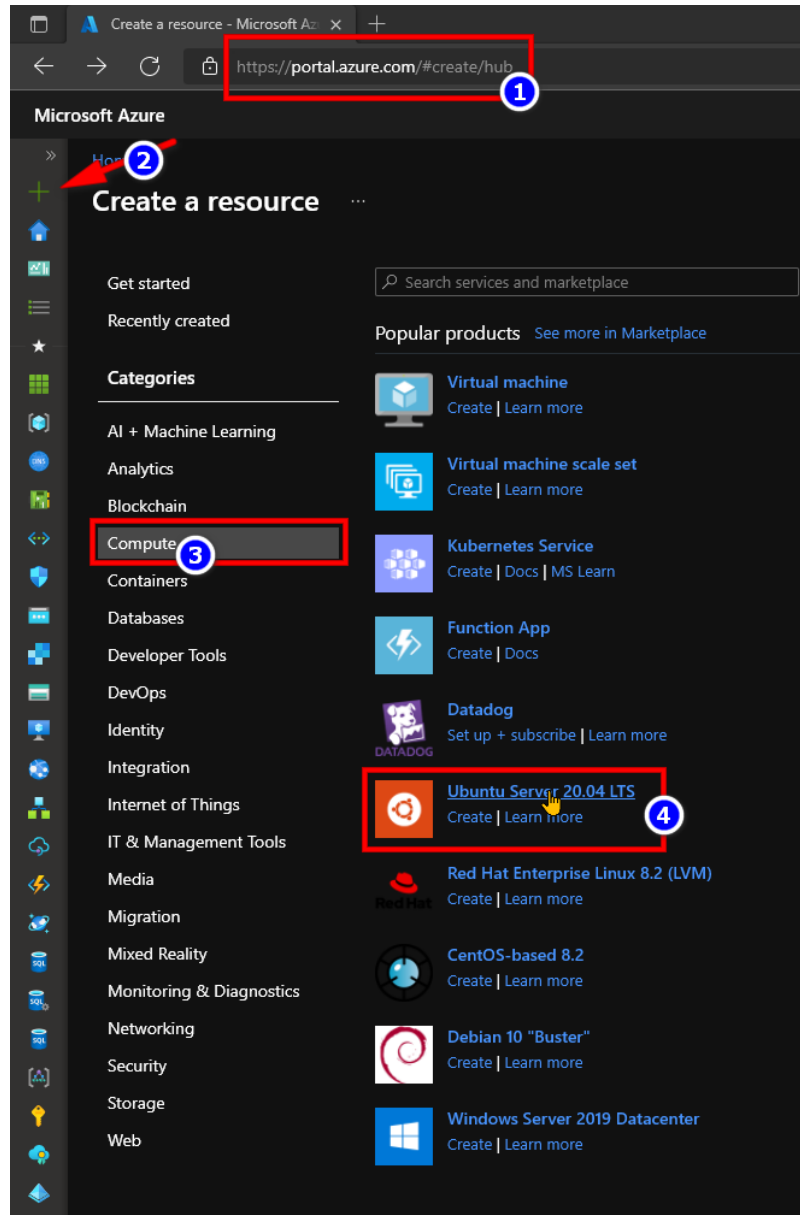
- ebook.sergiocabral.com/nodejs-logging-to-elasticsearch (versão on-line)
- ebook.sergiocabral.com/nodejs-logging-to-elasticsearch.pdf (versão para download em PDF)

1. Elasticsearch: Configurando ambiente

Será criada uma infraestrutura bem simples apenas para servir ao propósito da aula.

1.1. Criando o servidor

Vou começar criando uma Máquina Virtual (VM). Vou usar o Microsoft Azure como serviço de nuvem. Para mim é sempre mais fácil partir de uma imagem Linux Ubuntu.



Como qualquer banco de dados, o Elasticsearch consome armazenamento e memória. A memória é especialmente necessária ao realizar consultas e seu consumo é maior conforme mais dados em disco são gravados.

Mas para o propósito dessa aula pouca coisa é necessária. O motivo de eu escolher uma VM com 16GB de memória e 4 CPUs é porque 1) ela será temporária e 2) ao instalar as aplicações as coisas andam mais rápido.

Microsoft Azure

Home > Create a resource >

Create a virtual machine

Basics | Disks | Networking | Management | Advanced | Tags | Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group * [Create new](#)

Instance details

Virtual machine name *

Region *

Availability options

Security type

Image * [See all images](#) | [Configure VM generation](#)

Azure Spot instance ☐

Size * [See all sizes](#)

Administrator account

Authentication type ☐ SSH public key ☒ Password

Username *

Password *

Confirm password *

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ☐ None ☒ Allow selected ports

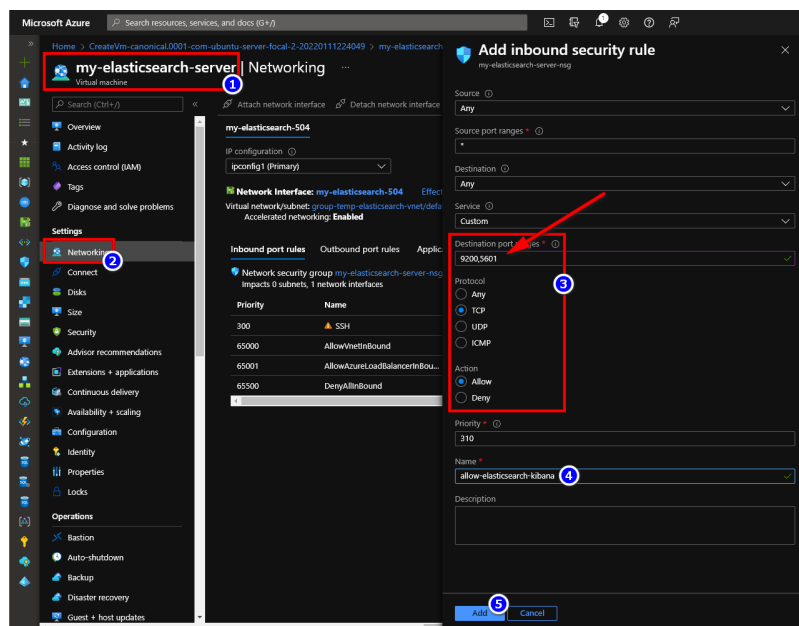
Select inbound ports *

Warning: This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

[Review + create](#) | < Previous | Next : Disks >

Então, simplesmente defini esses parâmetros e cliquei avançar até concluir.

Depois de criada a VM, precisamos acessá-la no painel do Azure para poder abrir as portas de rede (firewall) que vão permitir as conexões de entrada. As portas permitidas foram 9200 e 5601, que respectivamente são do Elasticsearch (o banco de dados) e Kibana (a interface web de visualização dos dados).



Como já disse, por ser uma VM temporária eu não fiz questão de levar em conta nenhuma boa prática de segurança ou seja lá o que for. Eu só quero um servidor Elasticsearch.

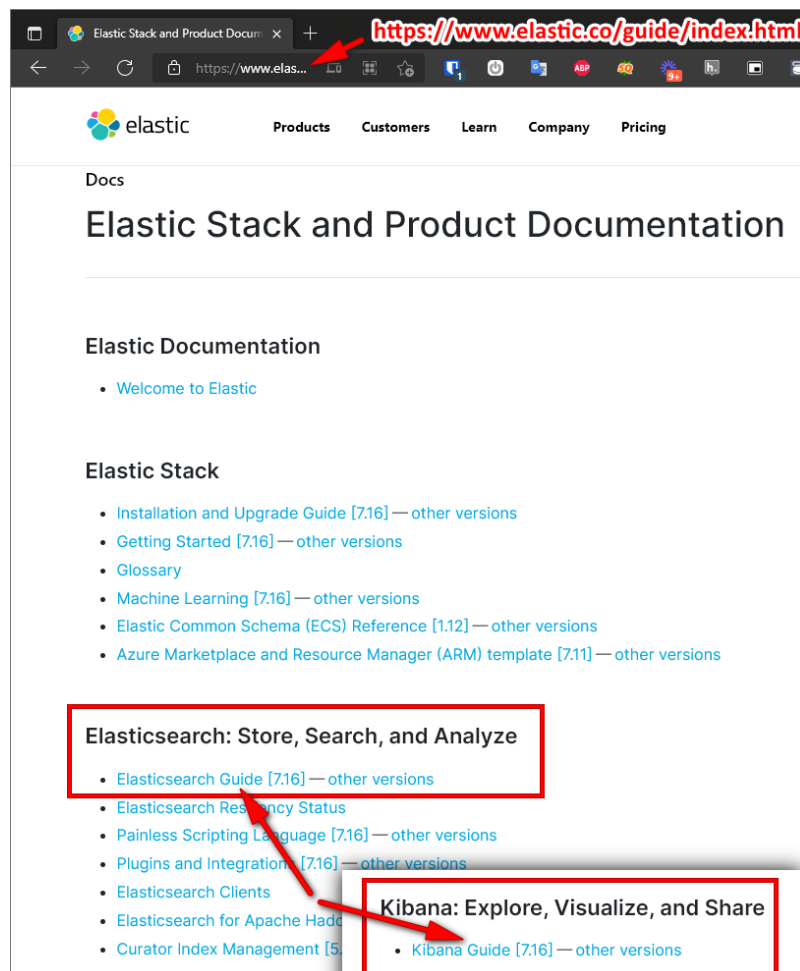
1.2. Configurando o servidor

Após a criação da VM executei os seguintes comandos para obter um ambiente do Elasticsearch e Kibana.

```
# Conectando na VM conforme o IP que o Azure atribuiu.  
# Usando o usuario e senha que escolhi.  
ssh adminer@20.206.112.142  
  
# Faz download da lista aplicacoes disponiveis para instalacao.  
sudo apt update  
  
# Instala o docker.  
sudo apt install docker.io -y
```

1.3. Instalando Elasticsearch e Kibana via Docker

No site oficial do Elasticsearch, em [elastic.co](https://www.elastic.co), opção de menu "Learn" e, então, "Docs", você acha os tutoriais de instalação.



Seguindo o [tutorial para instalar via Docker](#), em "Set up Elasticsearch", "Installing Elasticsearch" e "Install Elasticsearch with Docker". E consultando também o [tutorial para definir senha de autenticação](#) em "Secure the Elastic Stack", "Configuring security" e "Set up minimal security"...

Elasticsearch Guide

+ Elasticsearch Guide: 7.16 (current) ▾
+ What is Elasticsearch?
What's new in 7.16
Quick start
- Set up Elasticsearch
- Installing Elasticsearch
Install Elasticsearch from archive on Linux or MacOS
Install Elasticsearch with .zip on Windows
Install Elasticsearch with Debian Package
Install Elasticsearch with RPM
Install Elasticsearch with Windows MSI Installer
Install Elasticsearch with Docker 1
Install Elasticsearch on macOS with Homebrew
+ Snapshot and restore
- Secure the Elastic Stack
Elasticsearch security principles
- Configuring security
Set up minimal security 2
Set up basic security
Set up basic security plus HTTPS
Encrypting communications in an Elasticsearch Docker Container
Enabling cipher suites for stronger encryption
Supported SSL/TLS versions by JDK version
Security files
FIPS 140-2
+ Updating node security certificates

Os comandos necessários são:

```
# Cria uma rede que sera usada para o Kibana se comunicar com o Elasticsearch.
sudo docker network create elastic-network
```

```
# Executar uma imagem:      sudo docker run
# Em segundo plano:         -d
# Nome do container:        --name elasticsearch
# Nome da rede:             --net elastic-network
# Sempre reinicia o container: --restart=always
# Servidor unico:           -e "discovery.type=single-node"
# Habilita autenticacao:    -e "xpack.security.enabled=true"
# Define a senha tipo admin -e "ELASTIC_PASSWORD=my_password_for_elastic"
# Porta para dados via API:  -p 9200:9200
# Porta para dados binarios: -p 9300:9300
# Imagem Docker:            docker.elastic.co/elasticsearch/elasticsearch:7.16.2
#
```

```
sudo docker run -d --name elasticsearch --net elastic-network --restart=always -e
"discovery.type=single-node" -e "xpack.security.enabled=true" -e
"ELASTIC_PASSWORD=my_password_for_elastic" -p 9200:9200 -p 9300:9300
docker.elastic.co/elasticsearch/elasticsearch:7.16.2
```

```
# Executar uma imagem:      sudo docker run
# Em segundo plano:         -d
# Nome do container:        --name kibana
# Nome da rede:             --net elastic-network
# Sempre reinicia o container: --restart=always
# Endereco do servidor      -e "ELASTICSEARCH_HOSTS=http://elasticsearch:9200"
# Nome do usuario           -e "ELASTICSEARCH_USERNAME=elastic"
# Senha do usuario         -e "ELASTICSEARCH_PASSWORD=my_password_for_elastic"
# Porta para dados via API:  -p 5601:5601
# Imagem Docker:            docker.elastic.co/kibana/kibana:7.16.2
#
```

```
sudo docker run -d --name kibana --net elastic-network --restart=always -e
"ELASTICSEARCH_HOSTS=http://elasticsearch:9200" -e "ELASTICSEARCH_USERNAME=elastic" -e
"ELASTICSEARCH_PASSWORD=my_password_for_elastic" -p 5601:5601
docker.elastic.co/kibana/kibana:7.16.2
```

Tendo por saída no terminal:

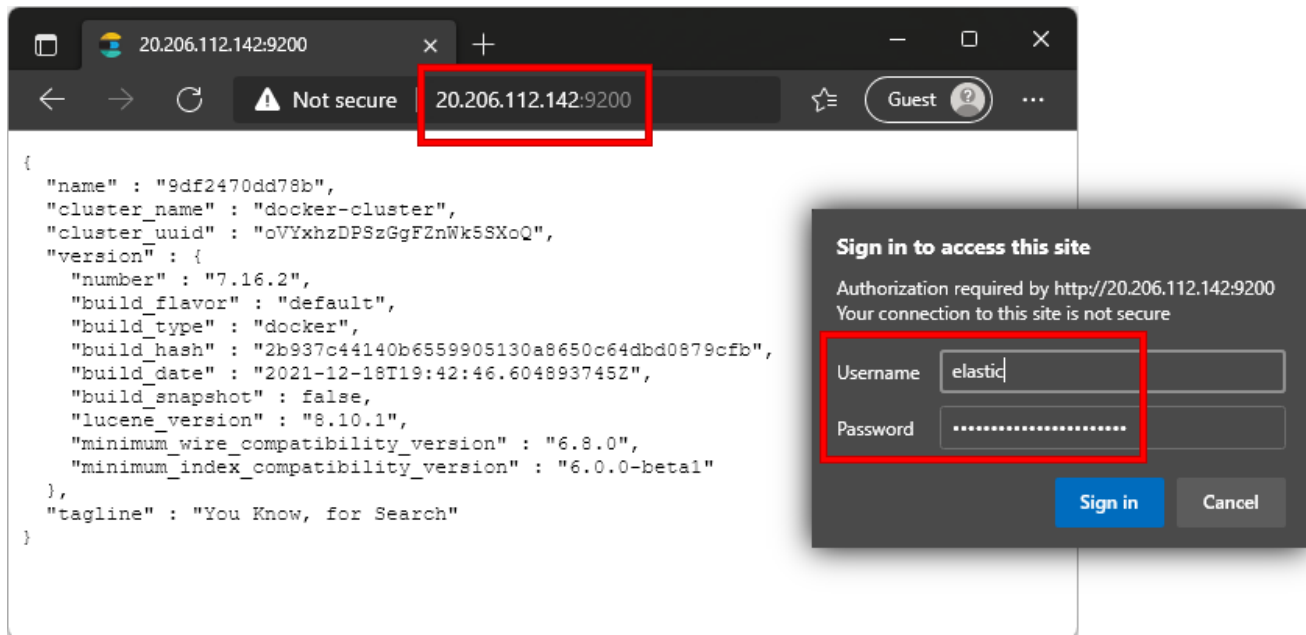
```
adminer@my-elasticsearch-serv x + v
adminer@my-elasticsearch-server:~$ sudo docker network create elastic-network
2b0c2b987aabed687b0eb3f62df0ad4f1c222abda41c8e5583f076b246f6814f
adminer@my-elasticsearch-server:~$ sudo docker run -d --name elasticsearch --net elastic-network --re
start=always -e "discovery.type=single-node" -e "xpack.security.enabled=true" -e "ELASTIC_PASSWORD=my_p
assword_for_elastic" -p 9200:9200 -p 9300:9300 docker.elastic.co/elasticsearch/elasticsearch:7.16.2
Unable to find image 'docker.elastic.co/elasticsearch/elasticsearch:7.16.2' locally
7.16.2: Pulling from elasticsearch/elasticsearch
da847062c6f6: Pull complete
f9947111a3a4: Pull complete
5f47506629dc: Pull complete
6728f6016cfb: Pull complete
3ee4bcac6dc4: Pull complete
cbb4caf74f49: Pull complete
60e3e554a3bf: Pull complete
64906e427669: Pull complete
96b7ea4c4a98: Pull complete
Digest: sha256:055ab3c3466c6bd72ef42f7773c5fa224db4fb7cd6a9a5588ebe46642a15abf5
Status: Downloaded newer image for docker.elastic.co/elasticsearch/elasticsearch:7.16.2
70d4b9d7bbf5926c8ad4d45731b05f2d3129270c6b29f78674c4b66e8bf3d9ac
adminer@my-elasticsearch-server:~$ sudo docker run -d --name kibana --net elastic-network --restart=al
ways -e "ELASTICSEARCH_HOSTS=http://elasticsearch:9200" -e "ELASTICSEARCH_USERNAME=elastic" -e "ELASTI
CSEARCH_PASSWORD=my_password_for_elastic" -p 5601:5601 docker.elastic.co/kibana/kibana:7.16.2
Unable to find image 'docker.elastic.co/kibana/kibana:7.16.2' locally
7.16.2: Pulling from kibana/kibana
22f1e563f2e0: Pull complete
64411cacc51b: Pull complete
aa7233e55da0: Pull complete
4a4f90843e5d: Pull complete
bf54180f96bb: Pull complete
bc16131a01d4: Pull complete
1853a5a39a21: Pull complete
6184c237acdd: Pull complete
3c322177b3fc: Pull complete
39fb241d872c: Pull complete
95b6fc095660: Pull complete
a84088a50be2: Pull complete
e69fac740d9d: Pull complete
Digest: sha256:9a83bce5d337e7e19d789ee7f952d36d0d514c80987c3d76d90fd1afd2411a9a
Status: Downloaded newer image for docker.elastic.co/kibana/kibana:7.16.2
098e476c8c8439dace872e08d3f6bc7e21d337d8436d095be5612097f2e3a574
adminer@my-elasticsearch-server:~$
```

1.4. Acessando as aplicações

As duas aplicações são servidas via protocolo **HTTP**. Então basta acessar o IP especificando a porta de cada aplicação.

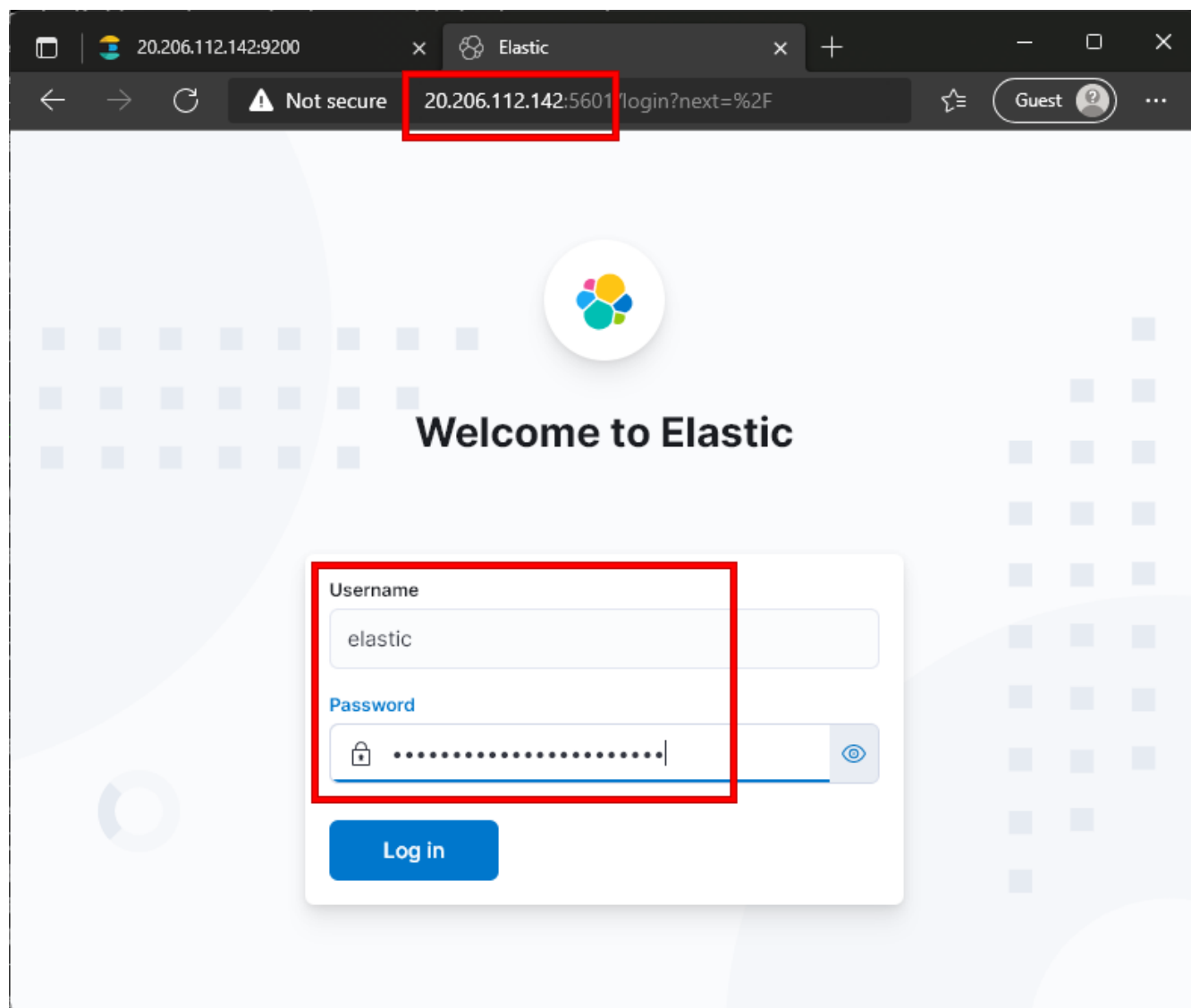
1.4.1. Elasticsearch

O Elasticsearch é servido por padrão na porta **9200**. Será solicitado usuário e senha, sendo o usuário tipo administrador **elastic** e a senha a que foi definida no passo anterior. Não existe interface porque é apenas um endpoint para receber requisições tipo API.



1.4.2. Kibana

O Kibana é de fato uma aplicação web, servida por padrão na porta **5601**. Use o mesmo usuário e senha mencionado no tópico anterior.



2. NodeJS: Criando uma aplicação base

O NodeJS é um ambiente para execução de JavaScript. Por outro lado, o NPM é um gerenciador de aplicações e bibliotecas feitas em NodeJS.

2.1. Criando uma aplicação tipo NPM

Para fazer pronto uso de um cliente capaz de se conectar ao banco de dados do Elasticsearch, começamos criando nossa aplicação como um pacote NPM e em seguida adicionamos a biblioteca [@elastic/elasticsearch](#), conforme orienta a [documentação oficial](#).

```
# Cria um projeto tipo NPM com parâmetros padrão
npm init -y

# Instala a biblioteca do cliente Elasticsearch
npm install @elastic/elasticsearch
```

Podemos ver mais sobre esse cliente acima e outros no site do Elasticsearch, em [elastic.co](#), opção de menu "Learn" e, então, "Docs".

Elasticsearch: Store, Search, and Analyze

- [Elasticsearch Guide \[7.16\]](#) — other versions
- [Elasticsearch Resiliency Status](#)
- [Painless Scripting Language \[7.16\]](#) — other versions
- [Plugins and Integrations \[7.16\]](#) — other versions
- [Elasticsearch Clients](#)
- [Elasticsearch for Apache Hadoop](#)
- [Curator Index Management](#)

Elasticsearch Clients

- [Java Client \[7.16\]](#) — other versions
- [JavaScript Client \[7.16\]](#) — other versions
- [Ruby Client \[7.16\]](#) — other versions
- [Go Client \[7.16\]](#) — other versions
- [.NET Clients \[7.16\]](#) — other versions
- [PHP Client \[7.16\]](#) — other versions
- [Perl Client \[master\]](#) — other versions
- [Python Client \[7.16\]](#) — other versions
- [eland](#)
- [Rust Client \[master\]](#) — other versions
- [Java REST Client \(deprecated\) \[7.15\]](#) — other versions
- [Java Transport Client \(deprecated\) \[7.16\]](#) — other versions
- [Community Contributed Clients](#)

2.2. Se comunicando com o Elasticsearch

Crie um arquivo chamado `index.js` e adicione o seguinte código:

```
const { Client } = require('@elastic/elasticsearch');

async function main() {
  // Conexão com o Elasticsearch
  const client = new Client({
    node: 'http://20.206.112.142:9200',
    auth: { username: 'elastic', password: 'my_password_for_elastic' }
  });

  // Inserir um documento
  await client.index({ index: 'my-index-name', id: '1234',
    body: { name: 'Sergio Cabral', website: 'sergiocabral.com' } });

  // Atualizar o documento anterior
  await client.update({ index: 'my-index-name', id: '1234',
    body: { doc: { website: 'sergiocabral.dev' } } });

  // Consulta o documento pelo id
  let response = await client.get({ index: 'my-index-name', id: '1234' });
  console.log(response.body);

  response = await client.search({ index: 'my-index-name', body: { query: {
    wildcard: { website: '*sergio*' } } } });
  console.log(response.body.hits.hits[0]);

  // Exclui o documento anterior pelo id
  await client.delete({ index: 'my-index-name', id: '1234' });
}

main();
```

Para mais possibilidades consulte a [documentação completa](#).