# ArangoDB

tutorialspoint

SIMPLY EASY LEARNING

## About the Tutorial

Apparently, the world is becoming more and more connected. And at some point in the very near future, your kitchen bar may well be able to recommend your favorite brands of whiskey! This recommended information may come from retailers, or equally likely it can be suggested from friends on Social Networks; whatever it is, you will be able to see the benefits of using graph databases, if you like the recommendations.

This tutorial explains the various aspects of ArangoDB which is a major contender in the landscape of graph databases. Starting with the basics of ArangoDB which focuses on the installation and basic concepts of ArangoDB, it gradually moves on to advanced topics such as CRUD operations and AQL. The last few chapters in this tutorial will help you understand how to deploy ArangoDB as a single instance and/or using Docker.

## Audience

This tutorial is meant for those who are interested in learning ArangoDB as a Multimodel Database. Graph databases are spreading like wildfire across the industry and are making an impact on the development of new generation applications. So anyone who is interested in learning different aspects of ArangoDB, should go through this tutorial.

## Prerequisites

Before proceeding with this tutorial, you should have the basic knowledge of Database, SQL, Graph Theory, and JavaScript.

## Copyright & Disclaimer

# Table of Contents

# 1. ArangoDB – A Multi-Model First Database

ArangoDB is hailed as a native multi-model database by its developers. This is unlike other NoSQL databases. In this database, the data can be stored as documents, key/value pairs or graphs. And with a single declarative query language, any or all of your data can be accessed. Moreover, different models can be combined in a single query. And, owing to its multi-model style, one can make lean applications, which will be scalable horizontally with any or all of the three data models.

## Layered vs. Native Multi-Model Databases

In this section, we will highlight a crucial difference between native and layered multi-model databases.

Many database vendors call their product "multi-model," but adding a graph layer to a key/value or document store does not qualify as native multi-model.

With ArangoDB, the same core with the same query language, one can club together different data models and features in a single query, as we have already stated in previous section. In ArangoDB, there is no "switching" between data models, and there is no shifting of data from A to B to execute queries. It leads to performance advantages to ArangoDB in comparison to the "layered" approaches.

## The Need for Multimodal Database

Interpreting the [Fowler's] basic idea leads us to realize the benefits of using a variety of appropriate data models for different parts of the persistence layer, the layer being part of the larger software architecture.

According to this, one might, for example, use a relational database to persist structured, tabular data; a document store for unstructured, object-like data; a key/value store for a hash table; and a graph database for highly linked referential data.

However, traditional implementation of this approach will lead one to use multiple databases in the same project. It can lead to some operational friction (more complicated deployment, more frequent upgrades) as well as data consistency and duplication issues.

The next challenge after unifying the data for the three data models, is to devise and implement a common query language that can allow data administrators to express a variety of queries, such as document queries, key/value lookups, graphy queries, and arbitrary combinations of these.

By **graphy queries**, we mean queries involving graph-theoretic considerations. In particular, these may involve the particular connectivity features coming from the edges. For example, **ShortestPath**, **GraphTraversal**, and **Neighbors**.

Graphs are a perfect fit as data model for relations. In many real-world cases such as social network, recommendor system, etc., a very natural data model is a graph. It captures relations and can hold label information with each edge and with each vertex. Further, JSON documents are a natural fit to store this type of vertex and edge data.

## ArangoDB ─ Features

There are various notable features of ArangoDB. We will highlight the prominent features below:

- Multi-model Paradigm
- ACID Properties
- HTTP API

ArangoDB supports all popular database models. Following are a few models supported by ArangoDB:

- Document model
- Key/Value model
- Graph model

A single query language is enough to retrieve data out of the database.

The four properties **Atomicity**, **Consistency**, **Isolation**, and **Durability** (ACID) describe the guarantees of database transactions. ArangoDB supports ACID-compliant transactions.

ArangoDB allows clients, such as browsers, to interact with the database with HTTP API, the API being resource-oriented and extendable with JavaScript.

# 2. ArangoDB – Advantages

Following are the advantages of using ArangoDB:

## Consolidation

As a native multi-model database, ArangoDB eliminates the need to deploy multiple databases, and thus decreases the number of components and their maintenance. Consequently, it reduces the technology-stack complexity for the application. In addition to consolidating your overall technical needs, this simplification leads to lower total cost of ownership and increasing flexibility.

## Simplified Performance Scaling

With applications growing over time, ArangoDB can tackle growing performance and storage needs, by independently scaling with different data models. As ArangoDB can scale both vertically and horizontally, so in case when your performance demands a decrease (a deliberate, desired slow-down), your back-end system can be easily scaled down to save on hardware as well as operational costs.

## Reduced Operational Complexity

The decree of Polyglot Persistence is to employ the best tools for every job you undertake. Certain tasks need a document database, while others may need a graph database. As a result of working with single-model databases, it can lead to multiple operational challenges. Integrating single-model databases is a difficult job in itself. But the biggest challenge is building a large cohesive structure with data consistency and fault tolerance between separate, unrelated database systems. It may prove nearly impossible.

Polyglot Persistence can be handled with a native multi-model database, as it allows to have polyglot data easily, but at the same time with data consistency on a fault tolerant system. With ArangoDB, we can use the correct data model for the complex job.

## Strong Data Consistency

If one uses multiple single-model databases, data consistency can become an issue. These databases aren't designed to communicate with each other, therefore some form of transaction functionality needs to be implemented to keep your data consistent between different models.

Supporting ACID transactions, ArangoDB manages your different data models with a single back-end, providing strong consistency on a single instance, and atomic operations when operating in cluster mode.

## Fault Tolerance

It is a challenge to build fault tolerant systems with many unrelated components. This challenge becomes more complex when working with clusters. Expertise is required to deploy and maintain such systems, using different technologies and/or technology stacks. Moreover, integrating multiple subsystems, designed to run independently, inflict large engineering and operational costs.

As a consolidated technology stack, multi-model database presents an elegant solution. Designed to enable modern, modular architectures with different data models, ArangoDB works for cluster usage as well.

## Lower Total Cost of Ownership

Each database technology requires ongoing maintenance, bug fixing patches, and other code changes which are provided by the vendor. Embracing a multi-model database significantly reduces the related maintenance costs simply by eliminating the number of database technologies in designing an application.

## Transactions

Providing transactional guarantees throughout multiple machines is a real challenge, and few NoSQL databases give these guarantees. Being native multi-model, ArangoDB imposes transactions to guarantee data consistency.

# 3. ArangoDB – Basic Concepts & Terminologies

In this chapter, we will discuss the basic concepts and terminologies for ArangoDB. It is very important to have a knowhow of the underlying basic terminologies related to the technical topic we are dealing with.

The terminologies for ArangoDB are listed below:

- Document
- Collection
- Collection Identifier
- Collection Name
- Database
- Database Name
- Database Organization

From the perspective of data model, ArangoDB may be considered a document-oriented database, as the notion of a document is the mathematical idea of the latter. Document-oriented databases are one of the main categories of NoSQL databases.

The hierarchy goes like this: Documents are grouped into collections, and Collections exist inside databases.

It should be obvious that Identifier and Name are two attributes for the collection and database.

Usually, two documents (vertices) stored in document collections are linked by a document (edge) stored in an edge collection. This is ArangoDB's graph data model. It follows the mathematical concept of a directed, labeled graph, except that edges don't just have labels, but are full-blown documents.

Having become familiar with the core terms for this database, we begin to understand ArangoDB's graph data model. In this model, there exist two types of collections: document collections and edge collections. Edge collections store documents and also include two special attributes: first is the **_from** attribute, and the second is the **_to** attribute. These attributes are used to create edges (relations) between documents essential for graph database. Document collections are also called vertex collections in the context of graphs (see any graph theory book).

Let us now see how important databases are. They are important because collections exist inside databases. In one instance of ArangoDB, there can be one or many databases. Different databases are usually used for multi-tenant setups, as the different sets of data inside them (collections, documents, etc.) are isolated from one another. The default database **_system** is special, because it cannot be removed. Users are managed in this database, and their credentials are valid for all the databases of a server instance.

# 4. ArangoDB – System Requirements

In this chapter, we will discuss the system requirements for ArangoDB.

The system requirements for ArangoDB are as follows:

- A VPS Server with Ubuntu Installation
- RAM: 1 GB; CPU : 2.2 GHz

For all the commands in this tutorial, we have used an instance of Ubuntu 16.04 (xenial) of RAM 1GB with one cpu having a processing power 2.2 GHz. And all the arangosh commands in this tutorial were tested for the ArangoDB version 3.1.27.

## How to Install ArangoDB?

In this section, we will see how to install ArangoDB. ArangoDB comes pre-built for many operating systems and distributions. For more details, please refer to the ArangoDB documentation. As already mentioned, for this tutorial we will use Ubuntu 16.04x64.

The first step is to download the public key for its repositories:

```
#    wget         https://www.arangodb.com/repositories/arangodb31/
xUbuntu_16.04/Release.key
```

### Output

```
--2017-09-03      12:13:24--         https://www.arangodb.com/reposit
ories/arangodb31/xUbuntu_16.04/Release.key Resolving      www.arangodb.com
(www.arangodb.com)...    104.25.1 64.21,     104.25.165.21,
2400:cb00:2048:1::6819:a415,    ... Connecting      to     www.arangodb.com
(www.arangodb.com)|104.25. 164.21|:443...    connected. HTTP     request sent,
awaiting    response...  200    OK Length:    3924   (3.8K) [application/pgp-keys]
Saving      to:    ‘Release.key’

Release.key              100%[===================>]            3.83K       -.-
KB/s        in     0.001s

2017-09-03  12:13:25     (2.61  MB/s)  -      ‘Release.key’saved  [39 24/3924]
```

The important point is that you should see the **Release.key saved** at the end of the output.

Let us install the saved key using the following line of code:

```
# sudo apt-key add Release.key
```

## Output

```
OK
```

Run the following commands to add the apt repository and update the index:

```
# sudo apt-add-repository 'deb
https://www.arangodb.com/repositories/arangodb31/xUbuntu_16.04/ /'

# sudo apt-get update
```

As a final step, we can install ArangoDB:

```
# sudo apt-get install arangodb3
```

## Output

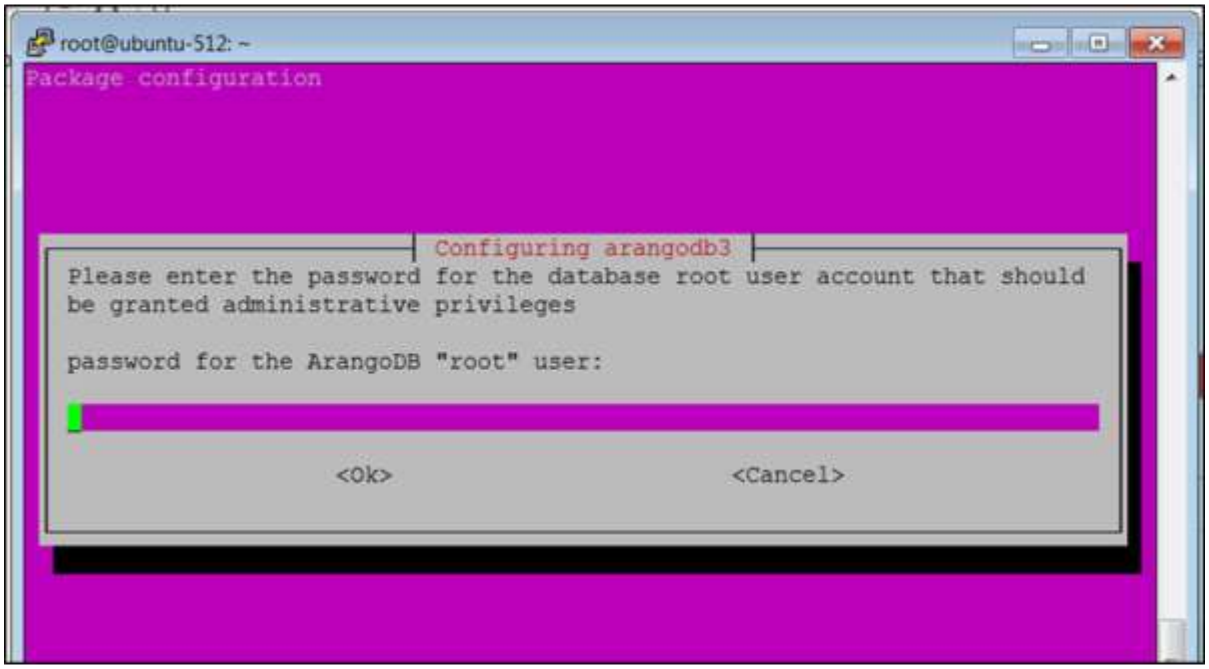
```
Reading package lists... Done

Building dependency tree

Reading state information... Done

The following package was automatically installed and is no longer required:

grub-pc-bin

Use 'sudo apt autoremove' to remove it.

The following NEW packages will be installed:

arangodb3

0 upgraded, 1 newly installed, 0 to remove and 17 not upgraded.

Need to get 55.6 MB of archives.

After this operation, 343 MB of additional disk space will be used.
```

Press **Enter**. Now the process of installing ArangoDB will start:

```
Get:1 https://www.arangodb.com/repositories/arangodb31/xUbuntu_16.04  arangodb3
3.1.27 [55.6 MB]

Fetched 55.6 MB in 59s (942 kB/s)

Preconfiguring packages ...

Selecting previously unselected package arangodb3.

(Reading database ... 54209 files and directories currently installed.)

Preparing to unpack .../arangodb3_3.1.27_amd64.deb ...

Unpacking arangodb3 (3.1.27) ...
```

```
Processing triggers for systemd (229-4ubuntu19) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up arangodb3 (3.1.27) ...
Database files are up-to-date.
```

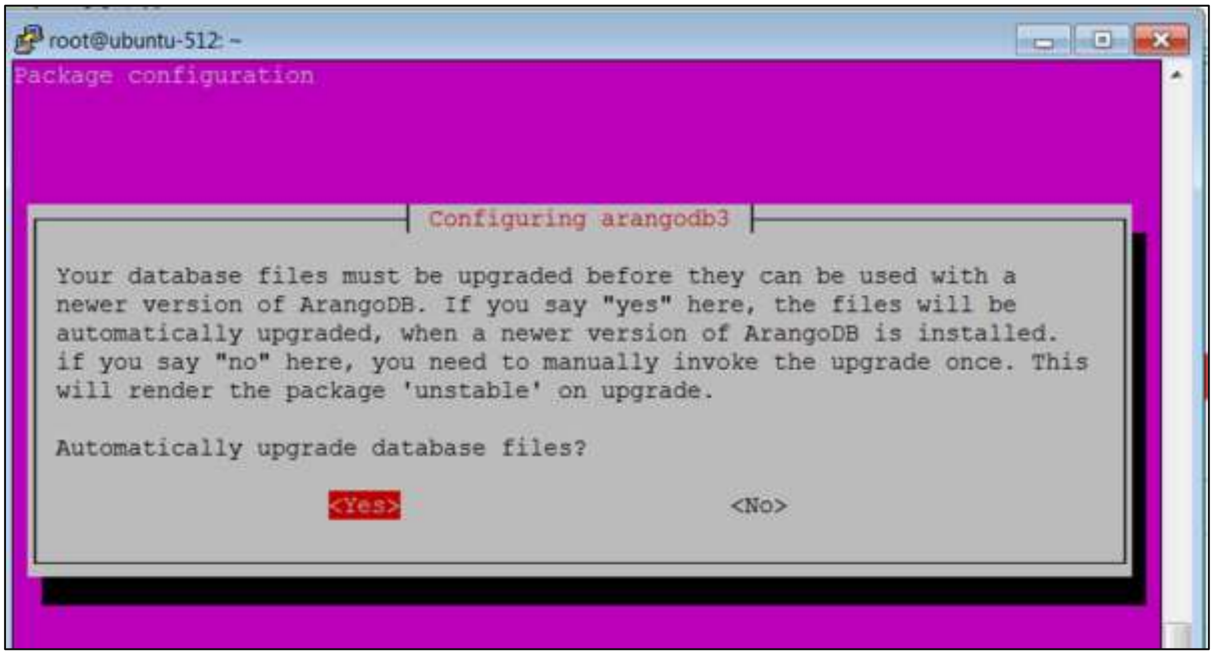
When the installation of ArangoDB is about to complete, the following screen appears:



```
Please enter the password for the database root user account that should
be granted administrative privileges

password for the ArangoDB "root" user:
```

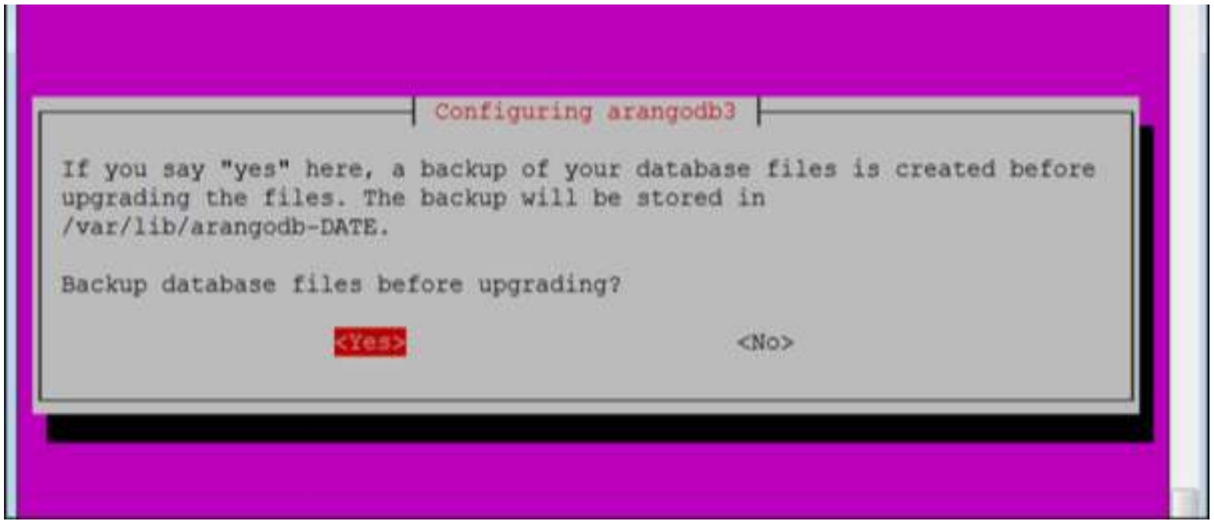
Here, you will be asked to provide a password for the ArangoDB **root** user. Note it down carefully.

Select the **yes** option when the following dialog box appears:



```
root@ubuntu-512: ~
Package configuration

                  ┤ Configuring arangodb3 ├

 Your database files must be upgraded before they can be used with a
 newer version of ArangoDB. If you say "yes" here, the files will be
 automatically upgraded, when a newer version of ArangoDB is installed.
 if you say "no" here, you need to manually invoke the upgrade once. This
 will render the package 'unstable' on upgrade.

 Automatically upgrade database files?

                <Yes>                          <No>
```

When you click **Yes** as in the above dialog box, the following dialog box appears. Click **Yes** here.



```
                  ┤ Configuring arangodb3 ├

 If you say "yes" here, a backup of your database files is created before
 upgrading the files. The backup will be stored in
 /var/lib/arangodb-DATE.

 Backup database files before upgrading?

                <Yes>                          <No>
```

You can also check the status of ArangoDB with the following command:

```
# sudo systemctl status arangodb3
```

## Output

```
arangodb3.service -      LSB:   arangodb
Loaded:     loaded (/etc/init.d/arangodb3;    bad;   vendor pre set:      enabled)
```

```
Active:     active (running)    since  Mon    2017-09-04   05:42:35     UTC;   4min
     46s    ago

Docs:man:systemd-sysv-generator(8)

Process:    2642   ExecStart=/etc/init.d/arangodb3  start  (co de=exited,
     status=0/SUC

Tasks:      22

Memory:     158.6M

CPU:  3.117s

CGroup:     /system.slice/arangodb3.service
                                  ├─2689      /usr/sbin/arangod   --uid  arangodb
     --gid  arangodb     --pid-file   /va
                                  └─2690      /usr/sbin/arangod   --uid  arangodb
     --gid  arangodb     --pid-file   /va

Sep  04     05:42:33     ubuntu-512   systemd[1]:  Starting     LSB:   aran
godb...

Sep  04     05:42:33     ubuntu-512   arangodb3[2642]:          *      Starting
     a rango      database     server a

Sep  04     05:42:35     ubuntu-512   arangodb3[2642]:    {startup}    sta rting
     up     in     daemon mode

Sep  04     05:42:35     ubuntu-512   arangodb3[2642]:    changed      working
     directory    for    child

Sep  04     05:42:35     ubuntu-512   arangodb3[2642]:
     ...done. Sep 04     05:42:35     ubuntu-512   systemd[1]:  Started      LSB:
     arang odb.

Sep  04     05:46:59     ubuntu-512   systemd[1]:  Started      LSB:   arang odb.
lines 1-19/19      (END)
```

ArangoDB is now ready to be used.

To invoke the arangosh terminal, type the following command in the terminal:

```
# arangosh
```

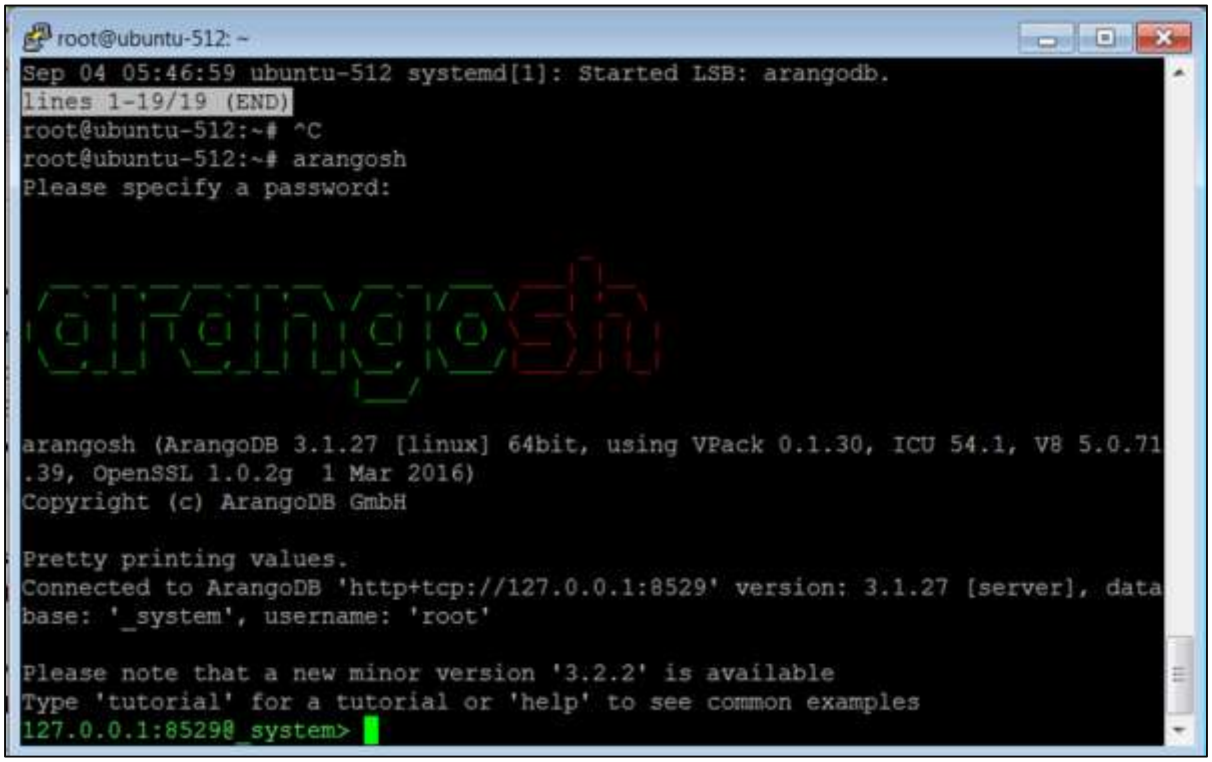
## Output

```
Please specify a password:
```

Supply the **root** password created at the time of installation:

```
_
__ _ _ _ __ __ _ _ _ __ __ _ ___ | |
/ | '__/ _ | ’ \ / ` |/ _ / | ’
| (| | | | (| | | | | (| | () _ \ | | |
_,|| _,|| ||_, |_/|/| ||
|__/
```

```
arangosh (ArangoDB 3.1.27 [linux] 64bit, using VPack 0.1.30, ICU 54.1, V8
5.0.71.39, OpenSSL 1.0.2g  1 Mar 2016)

Copyright (c) ArangoDB GmbH


Pretty printing values.

Connected to ArangoDB 'http+tcp://127.0.0.1:8529' version: 3.1.27 [server],
database: '_system', username: 'root'


Please note that a new minor version '3.2.2' is available

Type 'tutorial' for a tutorial or 'help' to see common examples

127.0.0.1:8529@_system> exit
```



```
Sep 04 05:46:59 ubuntu-512 systemd[1]: Started LSB: arangodb.
lines 1-19/19 (END)
root@ubuntu-512:~# ^C
root@ubuntu-512:~# arangosh
Please specify a password:

arangosh (ArangoDB 3.1.27 [linux] 64bit, using VPack 0.1.30, ICU 54.1, V8 5.0.71
.39, OpenSSL 1.0.2g  1 Mar 2016)
Copyright (c) ArangoDB GmbH

Pretty printing values.
Connected to ArangoDB 'http+tcp://127.0.0.1:8529' version: 3.1.27 [server], data
base: '_system', username: 'root'

Please note that a new minor version '3.2.2' is available
Type 'tutorial' for a tutorial or 'help' to see common examples
127.0.0.1:8529@_system>
```

To log out from ArangoDB, type the following command: