Learning Heroku Postgres

This book establishes a good learning mechanism by initially covering simple topics such as Heroku Toolbelt before moving on to complex topics such as data caching and tuning and PostGIS.

The book begins by giving you an introduction to the working of Heroku Postgres. This is followed by an overview of the PostgreSQL add-on, using which you'll create your first database and connect it using various programming languages such as Java, Ruby, Python, and Node.js. You will then look at securing your data, monitoring, working with permissions, and generating backups along with database rollbacks, followers, and forks to manage any loss in your data. Finally, you will improve the performance of your database by diving into log statements and understand how to avoid common mistakes.

Who this book is written for

Learning Heroku Postgres is targeted at developers and database admins. Even if you're new to Heroku Postgres, you'll be able to master both the basic as well as advanced features of Heroku Postgres. Since Heroku Postgres is incredibly user-friendly, no previous experience in computer coding or programming is required.

What you will learn from this book

- Introduce yourself to the inner workings of Heroku Postgres
- Deploy your first application with Heroku using the Heroku Toolbelt
- Work with permission levels to connect your database with a number of programming languages
- Secure your database using Postgres backups
- Interact with your data and share it with Dataclips
- Manage your data loss by performing rollbacks and working with followers and forks
- Identify common errors by analyzing logs and viewing metrics
- Add functionalities to your database with extensions such as PostGIS and Full Text Search Dictionaries

Patrick Espake

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Getting Started with Heroku Postgres'
- A synopsis of the book’s content
- More information on Learning Heroku Postgres

About the Author

Patrick Espake is passionate about technology, innovation, software development, and entrepreneurship. He has been professionally building and deploying web applications for over 10 years; with Heroku, it has been over 5 years, and he has worked on amazing projects for companies across the world. Currently, he is a cofounder of http://coursify.me. When Patrick isn't coding or writing, you will usually find him traveling to some wonderful place with his family.

I would like to thank my parents for their dedication and advice during my life, especially my mother, Elza, for not sparing effort and for her sacrifices to educate me and turn me into the person I am today. I would say that all your efforts were worthwhile and finishing this book together was a great achievement.

I also could never have finished this book without the support and encouraging words from my wife, Priscila. Through her infinite and pure love, she encouraged me and showed enormous patience during the several months that I've dedicated to this book. I would like to say that I am very grateful to be your husband.

I would also like to send a huge thank you to the Bard Kunenn family, who invited me numerous times for lunch during the weekends that I spent writing this book and provided me countless encouraging words for the creation of this book.

Finally, I would like to say thank you very much to my friend, Ana Carolina, for helping me in reviewing this book and for being so helpful with her suggestions for improvement.
Learning Heroku Postgres

This book is a definitive guide on how to use PostgreSQL on Heroku. Learn how to work with backups, dataclips, rollback, followers, forks, extensions, PostGIS, data caching, tuning, log statements, and common errors.

Heroku Postgres allows you to manage your PostgreSQL databases in a simple, worry-free way and from anywhere. Through the offered features, you can easily scale your database and extend functionalities.

This book is suitable for all Heroku Postgres user levels and offers knowledge that will help you manage your database with tranquility. The book covers simple topics until we reach more complex issues such as data caching and tuning. Through this book, you will be able to work with all the functionalities provided by Heroku Postgres.

What This Book Covers

Chapter 1, Getting Started with Heroku Postgres, introduces the Heroku architecture and how to build applications using Heroku Postgres. It also describes the key concepts about Heroku.

Chapter 2, Heroku Toolbelt, describes the Heroku Toolbelt and all the necessary tools to get started using Heroku at the command-line.

Chapter 3, Postgres Add-on, covers how to configure your local development environment, how to use the add-on, how to create your database, and how to connect using different programming languages. It also introduces the concepts of monitoring and information logs.

Chapter 4, PG Backups, introduces how to generate backups, import/export databases, data security, continuous protection, and how to upgrade your database plan.

Chapter 5, Dataclips, describes how dataclips work and how to share them with your colleagues. It also provides knowledge about security and limitations.

Chapter 6, Rollback, Followers, and Forks, covers important concepts related to security, stability, and experiments in using PostgreSQL database on the Heroku platform. It discusses topics related to rollback, followers, and forks.

Chapter 7, Understanding Log Statements and Common Errors, introduces how to view logs, how to collect metrics, and how to understand the most common errors.

Chapter 8, Extensions, PostGIS, Full Text Search Dictionaries, Data Caching, and Tuning, describes a collection of advanced features: how to install the main extensions of the Postgres database, how to enable PostGIS for work with spatial data, how to make optimizations, the text search tools, and the data cache.
Appendix A, *Keyword List*, contains a set of keywords used in this book, along with their definitions to facilitate better understanding of the concepts.

Appendix B, *Self-test Answers*, contains the answers to questions in each chapter that help test the knowledge you acquired.
Heroku simplifies the infrastructure of web applications. With its architecture, it is possible to create robust, manageable, and scalable applications according to the needs of your business.

This book is focused on how Heroku works with PostgreSQL. It will cover pieces of architecture and how to build applications that benefit from Heroku Postgres. In this first chapter, you will learn about the key concepts of Heroku, how it works, what are the supported versions of PostgreSQL, how to choose the best plan according to your need, and how high availability works. The concepts covered in this chapter are fundamental to understanding the other chapters.

In this chapter, will cover the following topics:

- How does Heroku work?
- Postgres supported versions
- Choosing the right Heroku Postgres plan
- Production-tier technical characterization
- High availability

**How does Heroku work?**

Heroku is a multi-language cloud platform that enables you to deploy applications written in several programming languages such as Ruby, Java, Python, Clojure, Scala, and Node.js. The list of supported programming languages is always growing.
The main idea of Heroku is to take away the pain of managing and scaling servers, so you can focus on the development of your product and deliver more functionality to your client, where Heroku is responsible for the infrastructure.

You don't need knowledge of servers to build a robust application with millions of users around the world.

**Heroku Dashboard and Heroku Toolbelt**

Heroku has a dashboard where you can easily manage your application. Through this dashboard, you can manage your resources, dynos, workers, add-ons, metrics, activity log, access permissions, and settings. In this chapter, you will learn to better understand these items.
Heroku also provides a command-line client to manage your applications. This client is called Heroku Toolbelt and is available at https://toolbelt.heroku.com. It is available for Mac OS X, Windows, Debian/Ubuntu, or in standalone mode. Heroku Toolbelt is a set of tools for managing and displaying the information about your applications. Most developers prefer to use Heroku Toolbelt instead of Heroku Dashboard because of the command-line facilities, in addition to offering a broader set of information. In the following chapter, you will understand how it works.

Deploying your applications

The first item you need to understand is how the deploy process happens. An application is a collection of code with its associated dependencies. In the case of Ruby on Rails, these dependencies are found in Gemfile in Python requirements.txt, in Node.js package.json, and in Java pom.xml. This gives further information on how Heroku should work with your code.
Getting Started with Heroku Postgres

If you use Ruby, Node.js, Python, or Java, Heroku can easily identify how to make your code executable, through the conventions that these languages adopt. For example, in the case of Ruby on Rails, Heroku knows it needs to run the Rails server to run your application.

If Heroku cannot run your application, you need to show how to do it. You must do so through the creation of a `Procfile` in your project, which contains instructions on how to run your application, as follows:

```bash
web: java -jar lib/my_app.jar $PORT
```

**Downloading the example code**

You can download the example code files from your account at http://www.packtpub.com for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

Heroku has adopted a very simple way to send your code to deploy. Each application built on Heroku has a Git repository, you should do a `git push heroku master` to start the deploy process. In the next chapter, you will deploy your first application and understand it better.

Heroku handles your code and dependencies by generating an optimized slug of your application. This slug is placed in each dyno that you have and each slug contains a compressed copy of your application. The slug is created by the slug compiler and its core is a collection of scripts called a buildpack. For each language supported by Heroku, there is one buildpack available. You can find more information about each of them in their repositories on GitHub:

- **Clojure**: https://github.com/heroku/heroku-buildpack-clojure
- **Gradle**: https://github.com/heroku/heroku-buildpack-gradle
- **Grails**: https://github.com/heroku/heroku-buildpack-grails
- **Java**: https://github.com/heroku/heroku-buildpack-java
- **Node.js**: https://github.com/heroku/heroku-buildpack-nodejs
- **PHP**: https://github.com/heroku/heroku-buildpack-php
- **Play**: https://github.com/heroku/heroku-buildpack-play
- **Python**: https://github.com/heroku/heroku-buildpack-python
- **Ruby**: https://github.com/heroku/heroku-buildpack-ruby
- **Scala**: https://github.com/heroku/heroku-buildpack-scala

The dyno manager is responsible for managing each dyno with your application slug.
Heroku architecture

Heroku's architecture is very robust and equipped to work with small applications such as blogs, personal websites, and large applications with a lot of users, such as Ello and Toyota.

This architecture serves the entire operation flow of its application, providing the mechanism to deploy your code until the management and scalability of your application.

The following diagram shows this architecture:
HTTP routing
The Heroku platform receives HTTP requests coming from the entry point herokuapp.com or by your custom domain, and forwards these requests to the load balancer, then to the routing mesh that distributes it for each dyno. The routing mesh is a personalized solution developed in Erlang based on MochiWeb (https://github.com/mochi/mochiweb). Each dyno has its own queue of requests to be processed. When a dyno is available, it picks up a request from its queue and the application inside the dyno processes it.

Dyno Manager
The Dyno Manager is responsible for maintaining the dynos running. When some dyno presents a problem, it is automatically replaced by a new dyno and the application information is loaded through the slug generated during the deploy. Because of the Dyno Manager, you do not need to make any changes to the operating system or other internal settings. Everything is done in an automated way.

Config vars
The traditional approach is setting the configuration variables in the source code of your application; this approach is not useful, especially when you have many environments such as production, development, and test. The safer approach is creating environment variables. Heroku works with environment variables called config vars.
Config vars are necessary to run your application. Some of these variables are provided by Heroku and others are provided by add-ons that you have installed. You can also add your own config vars. For example, for Postgres add-on the config vars are provided credentials to access the database. All the provided dynos have the same set of config vars. These variables are copied automatically when you create a new dyno.

**Understanding the Dynos**

A dyno is a single virtualization in a Unix container. This container contains the slug of your application created during the deploy and you can add more dynos at anytime for scaling your application. The dyno does not persist changes in the filesystem and does not contain a database. To save files, you need to use a shared service such as Amazon AWS S3 (Simple Storage Service), and use the data store's add-ons such as Heroku Postgres, ClearDB MySQL Database, MongoLab (MongoDB), Redis Cloud, GrapheneDB, or others to share a database data between your dynos. You can see all the available data stores' add-ons at [https://addons.heroku.com/#data-stores](https://addons.heroku.com/#data-stores).
Getting Started with Heroku Postgres

Workers
Workers are responsible for running background jobs in your applications hosted on Heroku. This is a key feature for building really scalable web applications. Through the workers, your applications can resize images, perform data upload to other servers, fetch API data, send e-mail, and perform many other tasks in the background. Your web requests always return immediately improving the user experience. On Heroku, the workers are scalable, so you can increase or decrease the quantity at anytime.

Add-ons
Heroku has a marketplace of add-ons. The idea of these add-ons is to provide integration with third-party services simply and seamlessly into your applications hosted on Heroku. Currently, there is a wide variety of add-ons that are divided in categories: Data Stores, Mobile, Search, Logging, e-mail, SMS, Workers and Queueing, Analytics, Caching, Monitoring, Media, Utilities, and Payments. Add-ons include Heroku Postgres, PG Backups, MemCachier, AirBrake, New Relic APM, Logentries, Deploy Hooks, Heroku Scheduler, Zerigo DNS, SSL, Websolr, SendGrid, Mandrill by MailChimp, and Codeship. You can meet these add-ons at https://addons.heroku.com/.

PostgreSQL is provided on Heroku via an add-on called Heroku Postgres. This add-on has a number of plans that vary in their features such as cache size, storage limit, limit of simultaneous connections, and the ability to work with forks and followers. In the Choosing the right Heroku Postgres plan section, you will understand more about these differences.

Logplex
Logplex is a distributed log collector, able to merge and redistribute multiple input streams to individual subscribers. Logs are collected from your app and other components of the Heroku platform. You can query this information through the Heroku Toolbelt or using the public API Logplex. These logs are collected from all the dynos and components that you have on Heroku.

Heroku API
The Heroku platform API allows you to create automated tasks, such as creating applications, collecting information from applications, managing add-ons, or performing other tasks programmatically, which previously could only be done through the Heroku dashboard. For example, Heroku Toolbelt uses Heroku API to perform its tasks.
Postgres supported versions

PostgreSQL is a relational database, open source, and very powerful. It has over 20 years of development and has a proved and reliable architecture. It is commonly adopted in many applications because of the following advantages:

- An open-source SQL standard compliant RDBMS
- Strong community
- Strong third-party support
- Extensible
- Objective

PostgreSQL is provided on Heroku via an add-on. This add-on has a free initial plan, only for small applications such as websites or blogs and then is available at a given series of paid plans, according to your need.

Each year a new version of PostgreSQL is released and Heroku adopts the newest version as standard in a very short time. Heroku keeps the current version and the last three versions. The versions currently available are:

- 9.4 (beta)
- 9.3 (default)
- 9.2
- 9.1
- 9.0 (deprecated, support ended December 3, 2014)

This means that every three years you need to upgrade the version of PostgreSQL used in your application because a new version of PostgreSQL is available and the last one is deprecated.

Choosing the right Heroku Postgres plan

Heroku offers three lines of plans with some differences in characteristics, these changes were made to suit small applications to applications that have a large volume of data or need transactional control.

Choosing the best plan will depend on the needs of your application. At any time, you can upgrade your database plan and optimize your application with a very short period of downtime. You will see more information about this in Chapter 6, Rollback, Followers, and Forks.
The main variations between the plans are expected downtime supported:

- Hobby tier tolerates 4 hours of downtime and is used for small and simple applications.
- Standard tier tolerates 1 hour of downtime and is used for production applications.
- Premium tier tolerates 15 minutes of downtime and is used for production applications where uptime is important.

The uptime expectations are measured based on a 30-day month.

The following table shows the main differences between the plans offered by Heroku and their respective prices:

<table>
<thead>
<tr>
<th>Plan</th>
<th>Connection Limit</th>
<th>Row Limit</th>
<th>RAM</th>
<th>Storage</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hobby Dev</td>
<td>20</td>
<td>10,000</td>
<td>0 Bytes</td>
<td>No</td>
<td>Free</td>
</tr>
<tr>
<td>Hobby Basic</td>
<td>20</td>
<td>10,000,000</td>
<td>0 Bytes</td>
<td>No</td>
<td>$9/mo</td>
</tr>
<tr>
<td>Standard 0</td>
<td>120</td>
<td>Unlimited</td>
<td>1 GB</td>
<td>64 GB</td>
<td>$50/mo</td>
</tr>
<tr>
<td>Premium 0</td>
<td>120</td>
<td>Unlimited</td>
<td>1 GB</td>
<td>64 GB</td>
<td>$200/mo</td>
</tr>
<tr>
<td>Standard 2</td>
<td>400</td>
<td>Unlimited</td>
<td>3.5 GB</td>
<td>256 GB</td>
<td>$200/mo</td>
</tr>
<tr>
<td>Premium 2</td>
<td>400</td>
<td>Unlimited</td>
<td>3.5 GB</td>
<td>256 GB</td>
<td>$350/mo</td>
</tr>
<tr>
<td>Standard 4</td>
<td>500</td>
<td>Unlimited</td>
<td>15 GB</td>
<td>512 GB</td>
<td>$750/mo</td>
</tr>
<tr>
<td>Premium 4</td>
<td>500</td>
<td>Unlimited</td>
<td>15 GB</td>
<td>512 GB</td>
<td>$1200/mo</td>
</tr>
<tr>
<td>Standard 5</td>
<td>500</td>
<td>Unlimited</td>
<td>30 GB</td>
<td>1 TB</td>
<td>$1400/mo</td>
</tr>
<tr>
<td>Standard 6</td>
<td>500</td>
<td>Unlimited</td>
<td>60 GB</td>
<td>1 TB</td>
<td>$2000/mo</td>
</tr>
<tr>
<td>Premium 5</td>
<td>500</td>
<td>Unlimited</td>
<td>30 GB</td>
<td>1 TB</td>
<td>$2500/mo</td>
</tr>
<tr>
<td>Premium 6</td>
<td>500</td>
<td>Unlimited</td>
<td>60 GB</td>
<td>1 TB</td>
<td>$3500/mo</td>
</tr>
<tr>
<td>Standard 7</td>
<td>500</td>
<td>Unlimited</td>
<td>120 GB</td>
<td>1 TB</td>
<td>$3500/mo</td>
</tr>
<tr>
<td>Premium 7</td>
<td>500</td>
<td>Unlimited</td>
<td>120 GB</td>
<td>1 TB</td>
<td>$6000/mo</td>
</tr>
</tbody>
</table>

**Shared features**

All the plans share the following features:

- Data clips to share data and queries with others people securely
- SSL protection to access psql/libpq
- Postgres extensions
- Web UI interface to manage the database
• Unmodified versions of Postgres
• Database service fully managed for automatic health checks
• Write-ahead log (WAL) that ensures minimal data loss in case of catastrophic failure, storing every minute

Production-tier technical characterization

The Standard, Premium, and Enterprise plans are offered for production applications that require certain operating characteristics based on multitenancy architectures, CPU, RAM, and IO. These characteristics can be more easily understood with the help of the following table:

<table>
<thead>
<tr>
<th>Plan</th>
<th>vCPU</th>
<th>RAM</th>
<th>Disk Size</th>
<th>Connection Limit</th>
<th>PIOPs</th>
<th>Multitenant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard 0</td>
<td>0</td>
<td></td>
<td>64 GB</td>
<td>120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Premium 0</td>
<td>2</td>
<td>1 GB</td>
<td>64 GB</td>
<td>120</td>
<td>200</td>
<td>Yes</td>
</tr>
<tr>
<td>Standard 2</td>
<td>2</td>
<td>3.5 GB</td>
<td>256 GB</td>
<td>400</td>
<td>200</td>
<td>Yes</td>
</tr>
<tr>
<td>Premium 2</td>
<td>2</td>
<td>15 GB</td>
<td>512 GB</td>
<td>400</td>
<td>1000</td>
<td>No</td>
</tr>
<tr>
<td>Standard 4</td>
<td>4</td>
<td>30 GB</td>
<td>1 TB</td>
<td>500</td>
<td>2000</td>
<td>No</td>
</tr>
<tr>
<td>Premium 4</td>
<td>4</td>
<td>60 GB</td>
<td>1 TB</td>
<td>500</td>
<td>3000</td>
<td>No</td>
</tr>
<tr>
<td>Standard 6</td>
<td>8</td>
<td>120 GB</td>
<td>1 TB</td>
<td>500</td>
<td>4000</td>
<td>No</td>
</tr>
<tr>
<td>Premium 6</td>
<td>16</td>
<td>240 GB</td>
<td>1 TB</td>
<td>500</td>
<td>4000</td>
<td>No</td>
</tr>
<tr>
<td>Standard 7</td>
<td>32</td>
<td>240 GB</td>
<td>1 TB</td>
<td>500</td>
<td>4000</td>
<td>No</td>
</tr>
<tr>
<td>Premium 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All plans run a 64-bit architecture, which ensures the best performance for the internal operations of Postgres. Heroku Postgres runs on a virtual infrastructure provided by Amazon AWS EC2.

The vCPU is the number of virtual processors in the instance and RAM is the amount of memory used to data cache. The PIOPs are a measure of how many IO disk operations can be performed per second.
For applications that require a lot of writes, IO can be a critical point. The data sets should fit in RAM, which ensures high performance with lower values of IOPs (Input/Output operations per second).

Finally, multitenancy refers to the principle of software architecture where a software instance is a server serving multiple client organizations (tenants).

**High availability**

The plans of the Premium and Enterprise tier come with the High Availability feature. This feature involves a cluster database and a management system to increase availability in case of failure.

When the database fails, it is automatically replaced by another. To determine that a failure occurred, Heroku starts a series of checks. This process can take up to 2 minutes. Because of the continuous protection mechanism, Heroku minimizes data loss, usually to 160 MB or 10 minutes, whichever is less.

When the recovery process finishes, new access credentials are generated and the database URL is changed. Finally the Heroku application is restarted with the new access data to the database.

**Self-test Questions**

Answer true or false:

1. Is Heroku a multi-language platform?
2. Using Heroku, does the developer have many concerns about infrastructure?
3. Is Heroku Toolbelt the most powerful way to manage your apps on Heroku?
4. Is it possible to deploy applications on Heroku via FTP?
5. Is dyno a virtualization of a Unix container?
6. Does Heroku always support the latest 10 versions of PostgreSQL?
7. Do the plans of the Premium tier tolerate 30 minutes of downtime?
8. Do all plans of Heroku run a 64-bit architecture?
9. Do only Standard plans have the capability of High Availability?
10. When a new database is built upon failure access, are the credentials changed?
Summary

In this chapter, you have learned how Heroku works, how to transform your application into a slug with dependencies that can be distributed across multiple dynos, making your application scalable. You have also learned that by default Heroku supports the current version of PostgreSQL and the last three versions, and every three years you need to upgrade the version used in your application.

You have also learned about the structure of the Heroku plans and your characteristics, where the Hobby plans are more oriented for blogs or personal websites, while the Standard, Premium, and Enterprise plans are ideal for production applications. You understood that the downtime of each plan must be taken into consideration to choose the plan that is best for your business.

You also got knowledge about the characterization of the Standard, Premium, and Enterprise plans and that there are variations of plans based on the usage of vCPU, RAM, and IO.

Finally, you learned the High Availability feature that manages and monitors the application and database in case of a failure. When a new database is created, the credentials are changed.

In the next chapter, you will learn about the Heroku Toolbelt and all tools you need to get started with using Heroku at the command line.
Where to buy this book

You can buy Learning Heroku Postgres from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.