Community Experience Distilled

Mastering Ansible

Design, develop, and solve real world automation and orchestration needs by unlocking the automation capabilities of Ansible

Jesse Keating

[PACKT] PUBLISHING open source* community experience distilled

Mastering Ansible

Design, develop, and solve real world automation and orchestration needs by unlocking the automation capabilities of Ansible

Jesse Keating

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

Mastering Ansible

First published: November 2015

Production reference:1191115

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B32PB, UK.

ISBN 978-1-78439-548-3

www.packtpub.com

Credits

Author
Jesse Keating

Reviewers
Ryan Eschinger
Sreenivas Makam
Tim Rupp
Sawant Shah
Patrik Uytterhoeven

Acquisition Editor
Meeta Rajani

Content Development Editor
Zeeyan Pinheiro

Technical Editor
Rohan Uttam Gosavi

Copy Editor
Pranjali Chury

Project Coordinator
Nidhi Joshi

Proofreader
Safis Editing

Indexer
Monica Ajmera Mehta

Graphics
Disha Haria

Production Coordinator
Arvindkumar Gupta

Cover Work
Arvindkumar Gupta

About the Author

Jesse Keating is an accomplished Ansible user, contributor, and presenter. He has been an active member of the Linux and open source communities for over 15 years. He has first-hand experience with a variety of IT activities, software development, and large-scale system administration. He has presented at numerous conferences and meet-ups, and he has written many articles on a variety of topics.

His professional Linux career started with Pogo Linux as a Lead Linux Engineer handling many duties, including building and managing automated installation systems. For 7 years, Jesse served at the Fedora Release Engineer as a senior software engineer at Red Hat. In 2012, he joined Rackspace to help manage Rackspace's public Cloud product, where he introduced the use of Ansible for the large-scale automation and orchestration of OpenStack-powered Clouds. Currently, he is a senior software engineer and the OpenStack release lead at Blue Box, an IBM company, where he continues to utilize Ansible to deploy and manage OpenStack Clouds.

He has worked as technical editor on *Red Hat Fedora and Enterprise Linux 4 Bible*, *A Practical Guide to Fedora and Red Hat Enterprise Linux 4th Edition*, *Python: Create-Modify-Reuse*, and *Practical Virtualization Solutions: Virtualization from the Trenches*. He has also worked as a contributing author on *Linux Toys II*, and *Linux Troubleshooting Bible*. You can find Jesse on Twitter using the handle `@iamjkeating`, and find his musings that require more than 140 characters on his blog at `https://derpops.bike`.

Acknowledgment

I'd like to thank my wife—my partner in crime, my foundation, my everything. She willingly took the load of our family so that I could hide away in a dark corner to write this book. Without her, it would never have been done. She was also there to poke me, not so gently, to put down the distractions at hand and go write! Thank you Jessie, for everything. I'd like to to thank my boys too, Eamon and Finian, for giving up a few (too many) evenings with their daddy while I worked to complete one chapter or another. Eamon, it was great to have you so interested in what it means to write a book. Your curiosity inspires me! Fin, you're the perfect remedy for spending too much time in serious mode. You can always crack me up! Thank you, boys for sharing your father for a bit.

I'd also like to thank all my editors, reviewers, confidants, coworkers past and present, and just about anybody who would listen to my crazy ideas, or read a blurb I put on the page. Your feedback kept me going, kept me correct, and kept my content from being completely adrift.

About the Reviewers

Ryan Eschinger is an independent software consultant with over 15 years of experience in operations and application development. He has a passion for helping businesses build and deploy amazing software. Using tools such as Ansible, he specializes in helping companies automate their infrastructure, establish automated and repeatable deployments, and build virtualized development environments that are consistent with production. He has worked with organizations of all shapes, sizes, and technical stacks. He's seen it all—good and bad—and he loves what he does. You can find him in one of the many neighborhood coffee shops in Brooklyn, NY, or online at `http://ryaneschinger.com/`.

Sreenivas Makam is currently working as a senior engineering manager at Cisco Systems, Bangalore. He has a master's degree in electrical engineering and around 18 years of experience in the networking industry. He has worked on both start-ups and big established companies. His interests include SDN, NFV, Network Automation, DevOps, and Cloud technologies. He also likes to try out and follow open source projects in these areas. You can find him on his blog at `https://sreeninet.wordpress.com/`.

Tim Rupp has been working in various fields of computing for the last 10 years. He has held positions in computer security, software engineering, and most recently, in the fields of Cloud computing and DevOps.

He was first introduced to Ansible while at Rackspace. As part of the Cloud engineering team, he made extensive use of the tool to deploy new capacity for the Rackspace Public Cloud. Since then, he has contributed patches, provided support for, and presented on Ansible topics at local meetups.

He is currently stationed at F5 Networks, where he is involved in Solution development as a senior software engineer. Additionally, he spends time assisting colleagues in various knowledge-sharing situations revolving around OpenStack and Ansible.

I'd like to thank my family for encouraging me to take risks and supporting me along the way. Without their support, I would have never come out of my shell to explore new opportunities. I'd also like to thank my girlfriend for putting up with my angry beaver moments as I balance work with life.

Sawant Shah is a passionate and experienced full-stack application developer with a formal degree in computer science.

Being a software engineer, he has focused on developing web and mobile applications for the last 9 years. From building frontend interfaces and programming application backend as a developer to managing and automating service delivery as a DevOps engineer, he has worked at all stages of an application and project's lifecycle.

He is currently spearheading the web and mobile projects division at the Express Media Group—one of the country's largest media houses. His previous experience includes leading teams and developing solutions at a software house, a BPO, a non-profit organization, and an Internet startup.

He loves to write code and keeps learning new ways to write optimal solutions. He blogs his experiences and opinions regarding programming and technology on his personal website, `http://www.sawantshah.com`, and on Medium, `https://medium.com/@sawant`.You can follow him on Twitter, where he shares learning resources and other useful tech material at `@sawant`.

Patrik Uytterhoeven has over 16 years of experience in IT. Most of this time was spent on HP Unix and Red Hat Linux. In late 2012, he joined Open-Future, a leading open source integrator and the first Zabbix reseller and training partner in Belgium.

When he joined Open-Future, he gained the opportunity to certify himself as a Zabbix Certified trainer. Since then, he has provided training and public demonstrations not only in Belgium but also around the world, in countries such as the Netherlands, Germany, Canada, and Ireland. His next step was to write a book about Zabbix. *Zabbix Cookbook* was born in March 2015 and was published by *Packt Publishing*.

As he also has a deep interest in configuration management, he wrote some Ansible roles for Red Hat 6.x and 7.x to deploy and update Zabbix. These roles, and some others, can be found in the Ansible Galaxy at `https://galaxy.ansible.com/list#/users/1375`.

He is also a technical reviewer of *Learning Ansible* and the upcoming book, *Ansible Configuration Management, Second Edition*, both by Packt Publishing.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and readPackt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at`www.PacktPub.com`, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

Welcome to Mastering Ansible, your guide to a variety of advanced features and functionality provided by Ansible, which is an automation and orchestration tool. This book will provide you with the knowledge and skills to truly understand how Ansible functions at the fundamental level. This will allow you to master the advanced capabilities required to tackle the complex automation challenges of today and beyond. You will gain knowledge of Ansible workflows, explore use cases for advanced features, troubleshoot unexpected behavior, and extend Ansible through customization.

What this book covers

Chapter 1, *System Architecture and Design of Ansible*, provides a detailed look at the ins and outs of how Ansible goes about performing tasks on behalf of an engineer, how it is designed, and how to work with inventories and variables.

Chapter 2, *Protecting Your Secrets with Ansible*, explores the tools available to encrypt data at rest and prevent secrets from being revealed at runtime.

Chapter 3, *Unlocking the Power of Jinja2 Templates*, states the varied uses of the Jinja2 templating engine within Ansible, and discusses ways to make the most out of its capabilities.

Chapter 4, *Controlling Task Conditions*, describes the changing of default behavior of Ansible to customize task error and change conditions.

Chapter 5, *Composing Reusable Ansible Content with Roles*, describes the approach to move beyond executing loosely organized tasks on hosts to encapsulating clean reusable abstractions to applying the specific functionality of a target set of hosts.

Chapter 6, *Minimizing Downtime with Rolling Deployments*, explores the common deployment and upgrade strategies to showcase relevant Ansible features.

Chapter 7, *Troubleshooting Ansible*, explores the various methods that can be employed to examine, introspect, modify, and debug the operations of Ansible.

Chapter 8, *Extending Ansible*, discovers the various ways in which new capabilities can be added to Ansible via modules, plugins, and inventory sources.

What you need for this book

To follow the examples provided in this book, you will need access to a computer platform capable of running Ansible. Currently, Ansible can be run from any machine with Python 2.6 or 2.7 installed (Windows isn't supported for the control machine). This includes Red Hat, Debian, CentOS, OS X, any of the BSDs, and so on.

This book uses the Ansible 1.9.x series release.

Ansible installation instructions can be found at `http://docs.ansible.com/ansible/intro_installation.html`.

Who this book is for

This book is intended for Ansible developers and operators who have an understanding of the core elements and applications but are now looking to enhance their skills in applying automation using Ansible.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "When `ansible` or `ansible-playbook` is directed at an executable file for an inventory source, Ansible will execute that script with a single argument, `--list`."

A block of code is set as follows:

```
- name: add new node into runtime inventory
  add_host:
    name: newmastery.example.name
    groups: web
    ansible_ssh_host: 192.168.10.30
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "The first is an SSH feature, **ControlPersist**, which provides a mechanism to create persistent sockets when first connecting to a remote host that can be reused in subsequent connections to bypass some of the handshaking required when creating a connection."

> Warnings or important notes appear in a box like this.

> Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

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.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
System Architecture and Design of Ansible

This chapter provides a detailed exploration of the architecture and design of how **Ansible** goes about performing tasks on your behalf. We will cover basic concepts of inventory parsing and how the data is discovered, and then dive into playbook parsing. We will take a walk through module preparation, transportation, and execution. Lastly, we will detail variable types and find out where variables can be located, the scope they can be used for, and how precedence is determined when variables are defined in more than one location. All these things will be covered in order to lay the foundation for mastering Ansible!

In this chapter, we will cover the following topics:

- Ansible version and configuration
- Inventory parsing and data sources
- Playbook parsing
- Module transport and execution
- Variable types and locations
- Variable precedence

Ansible version and configuration

It is assumed that you have Ansible installed on your system. There are many documents out there that cover installing Ansible in a way that is appropriate for the operating system and version that you might be using. This book will assume the use of the Ansible 1.9.x version. To discover the version in use on a system with Ansible already installed, make use of the version argument, that is, either `ansible` or `ansible-playbook`:

```
2. jkeating@serenity: ~/src/mastery (zsh)
~/src/mastery> ansible-playbook --version
ansible-playbook 1.9.3
  configured module search path = None
~/src/mastery>
```

> Note that `ansible` is the executable for doing ad-hoc one-task executions and `ansible-playbook` is the executable that will process playbooks for orchestrating many tasks.

The configuration for Ansible can exist in a few different locations, where the first file found will be used. The search order changed slightly in version 1.5, with the new order being:

- `ANSIBLE_CFG`: This is an environment variable
- `ansible.cfg`: This is in the current directory
- `ansible.cfg`: This is in the user's home directory
- `/etc/ansible/ansible.cfg`

Some installation methods may include placing a **config** file in one of these locations. Look around to check whether such a file exists and see what settings are in the file to get an idea of how Ansible operation may be affected. This book will assume no settings in the `ansible.cfg` file that would affect the default operation of Ansible.

Inventory parsing and data sources

In Ansible, nothing happens without an inventory. Even ad hoc actions performed on localhost require an inventory, even if that inventory consists just of the localhost. The inventory is the most basic building block of Ansible architecture. When executing `ansible` or `ansible-playbook`, an inventory must be referenced. Inventories are either files or directories that exist on the same system that runs `ansible` or `ansible-playbook`. The location of the inventory can be referenced at runtime with the `-inventory-file` (`-i`) argument, or by defining the path in an Ansible config file.

Inventories can be static or dynamic, or even a combination of both, and Ansible is not limited to a single inventory. The standard practice is to split inventories across logical boundaries, such as *staging* and *production*, allowing an engineer to run a set of plays against their staging environment for validation, and then follow with the same exact plays run against the production inventory set.

Variable data, such as specific details on how to connect to a particular host in your inventory, can be included along with an inventory in a variety of ways as well, and we'll explore the options available to you.

The static inventory

The static inventory is the most basic of all the inventory options. Typically, a static inventory will consist of a single file in the `ini` format. Here is an example of a static inventory file describing a single host, `mastery.example.name`:

```
mastery.example.name
```

That is all there is to it. Simply list the names of the systems in your inventory. Of course, this does not take full advantage of all that an inventory has to offer. If every name were listed like this, all plays would have to reference specific host names, or the special `all` group. This can be quite tedious when developing a playbook that operates across different sets of your infrastructure. At the very least, hosts should be arranged into groups. A design pattern that works well is to arrange your systems into groups based on expected functionality. At first, this may seem difficult if you have an environment where single systems can play many different roles, but that is perfectly fine. Systems in an inventory can exist in more than one group, and groups can even consist of other groups! Additionally, when listing groups and hosts, it's possible to list hosts without a group. These would have to be listed first, before any other group is defined.