Vagrant and Splunk Development

Building Splunk Apps for Any Environment

Jason Rauen

October 2019

Splunk Usergroup Slack

Senior Lead Technologist | Booz Allen Hamilton

Badarsebard

Splunk Usergroup Slack

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only, and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

Building Splunk Apps

Should be easy right?

splunk> .conf19

How It Starts

Simple apps require simple solutions

But what happens when things get complicated?

- Distributed and clustered architectures
- Support for multiple architectures
- Multiple developers
- Operating system specific requirements
- Support for multiple operating systems

Past Solutions

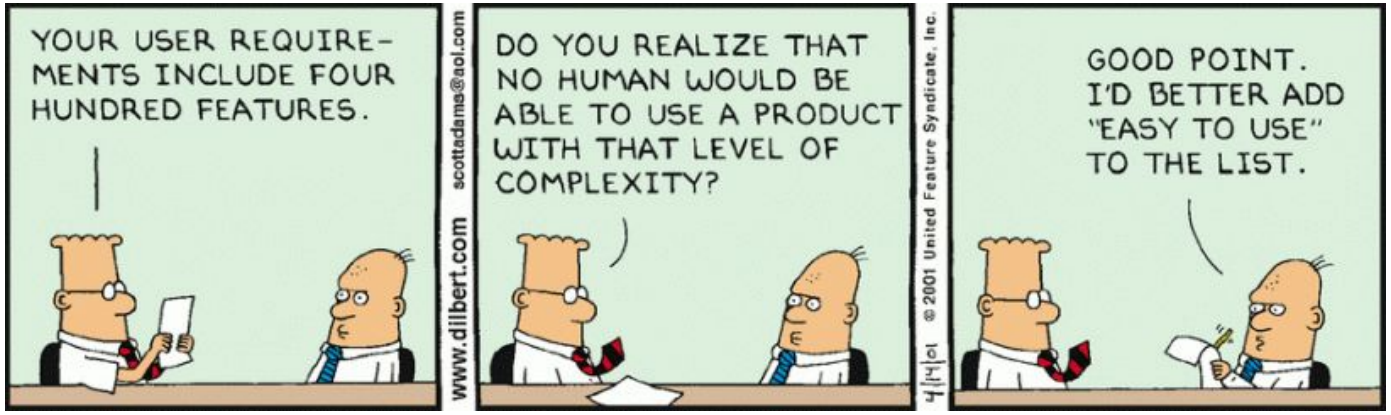
Create a remote environment for developers

Pros

- Access to the required operating system

Cons

- Mock deployments
  - For *each* architecture
- Multiple deployments for multiple team members
- Developer collisions
- Typically relies on another team for provisioning



Past Solutions

Multiple Splunk instances on a host

Pros

- Completely local solution
  - https://wiki.splunk.com/Community:Runmultiple_Splunks_on_one_machine

Cons

- Not supported
- Difficult
- Unfriendly to version control
- Running is a pain

Virtualization

Remote environments from the comfort of $HOME

- Run a virtualization tool like Virtualbox, Vmware, or Hyper-V
- Sweet! Problem solved. Time for cake.

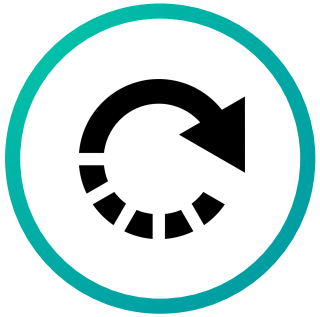
Except…

We need to do the following for each architecture

Create the virtual machine

Install and configure Splunk

Install and configure Splunk



Repeat for every server in every architecture

Document the process

Vagrant

Virtualization made easy

Enter Vagrant

Stage left

- Manage and run virtual machines simply
- Single file configuration for reproducible, portable, automated VMs
- Not a virtualization provider

Docker

A Contextual Sidebar

- Docker is a possible solution for some of the previously mentioned challenges.
- See the following .conf talks for examples:
- DE123416
- SF88089
- HL010
- FN118645

Goal

Vagrant as a Solution

- Docker is a good solution.
- In some circumstances Vagrant is better.
- Primary goal is to highlight Vagrant so developers have cho
- Why I used Vagrant for PyDen.

Basic Anatomy of Vagrant

Three steps to virtualization victory

Vagrant CLI

Vagrantfile

Base Box

Vagrant CLI

- A binary package you drop in your $PATH

Vagrantfile

- Ruby-like syntax configuration file
- Small and sharable through VCS

Base Box

- Shared starting image.
- Selected and modified by Vagrantfile.
- Public repository at app.vagrantup.com or share through corporate file sharing tools.

A Single Simple Workflow

Because ninjas are busy

splunk> .conf19

How to Splunk with Vagrant

- Vagrant provides us with a base virtual machine we can work on
- But how does it help us create a consistent deployment of Splunk from which to work?

Provisioners

Eureka!

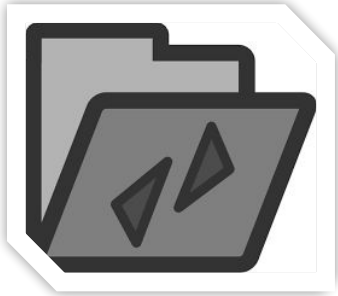
- Automated provisioning will install and configure software on the virtual machine.
- Some common provisioner tools available for use:
  - Shell
  - Ansible
  - CFEngine
  - Chef
  - Docker
  - Puppet
  - Salt
- Defined in the Vagrantfile.

Synced Folders

Easier than FTP

- Sync files between host and guest.
- Provider specific.
- Mapping defined in Vagrantfile.
- Default mapping between project root and /vagant on guest.

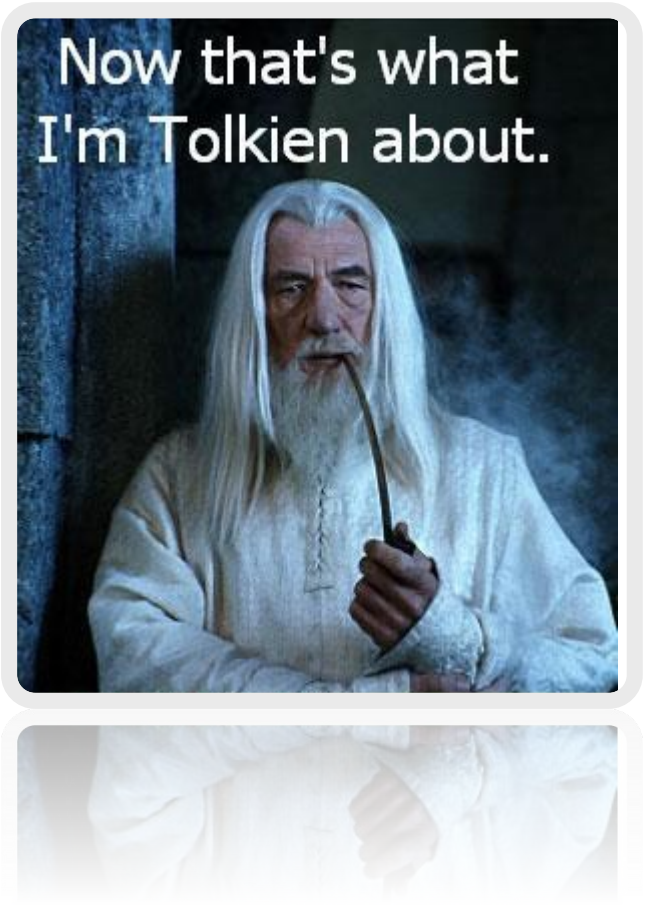





Vagrant and Multi-Machine Configs

One Config to Rule Them ALL

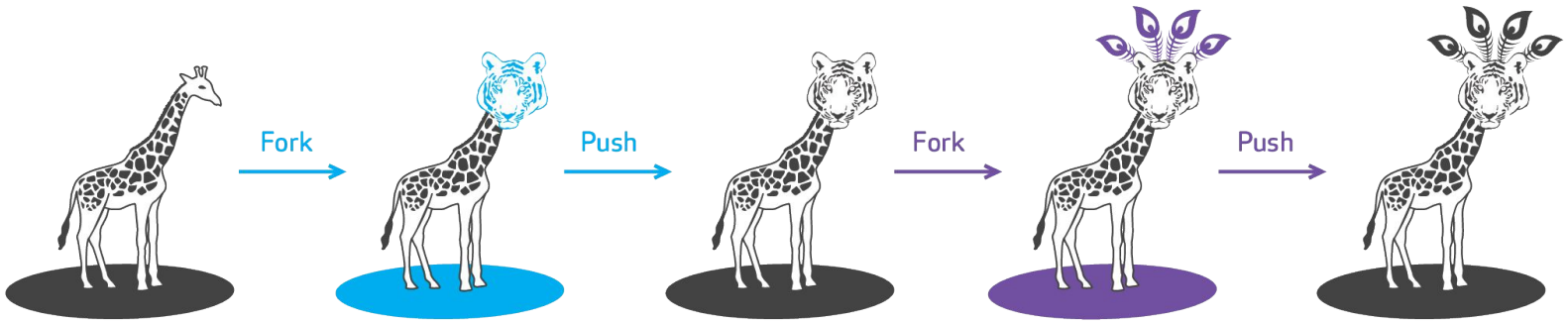
- Vagrant supports multi-machine environments
- Defined in the same Vagrantfile.
- Networking options available



Multiple Architectures

Building for all the things

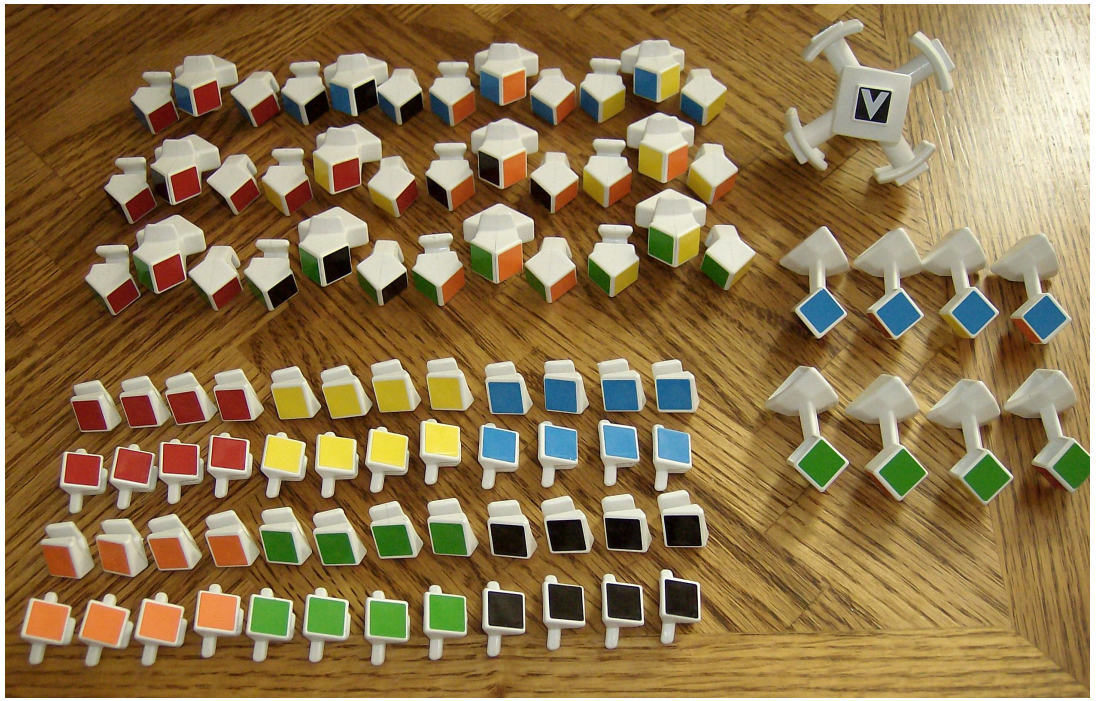
- Support for multiple architectures through multiple Vargantfiles
- No limit to Vagrantfile location
- Control particular architecture via changing working directory



Putting It All Together

The pieces fit!

- Single binary and config file
- Single Configuration for:
  - Creating and provisioning machines
  - Syncing files between them and the host
  - Networking requirements
  - Multiple machine architectures
- Multiple Configurations for multiple architectures



Developer Workflow

A day in the life of a Developer with Vagrant

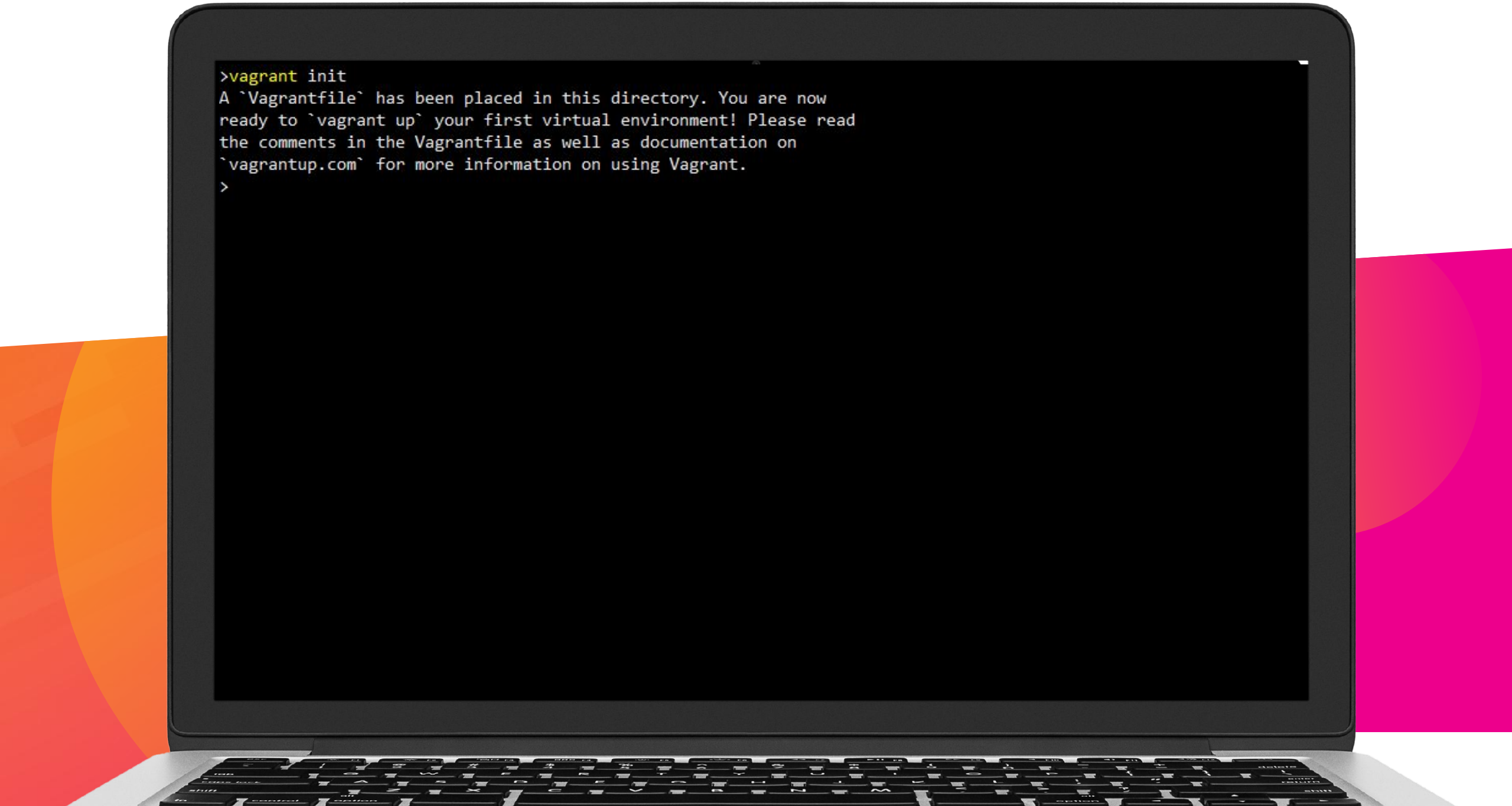
Initial

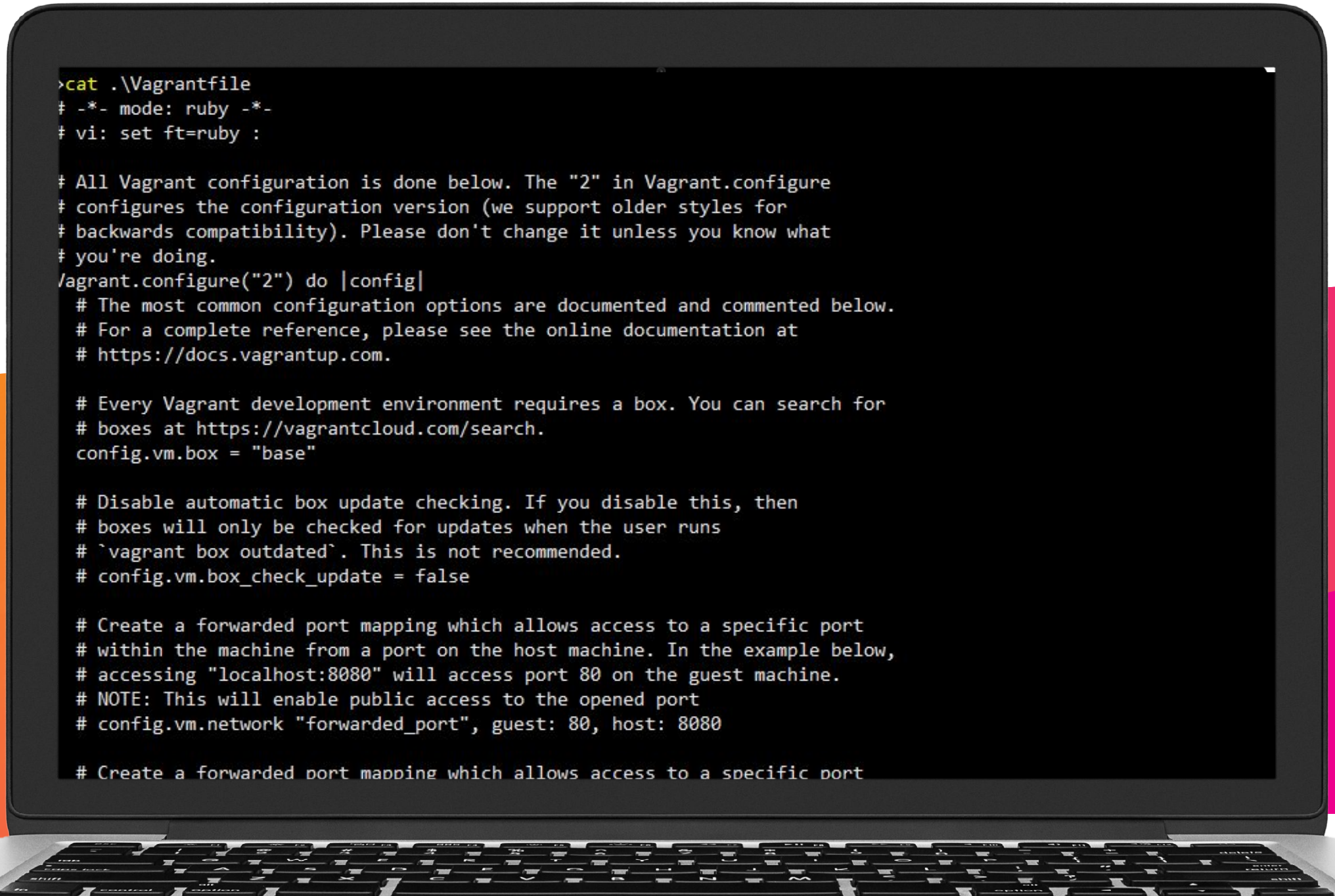
- Download Vagrant
- Clone repository or initialize Vagrant
- Install preferred virtualization provider

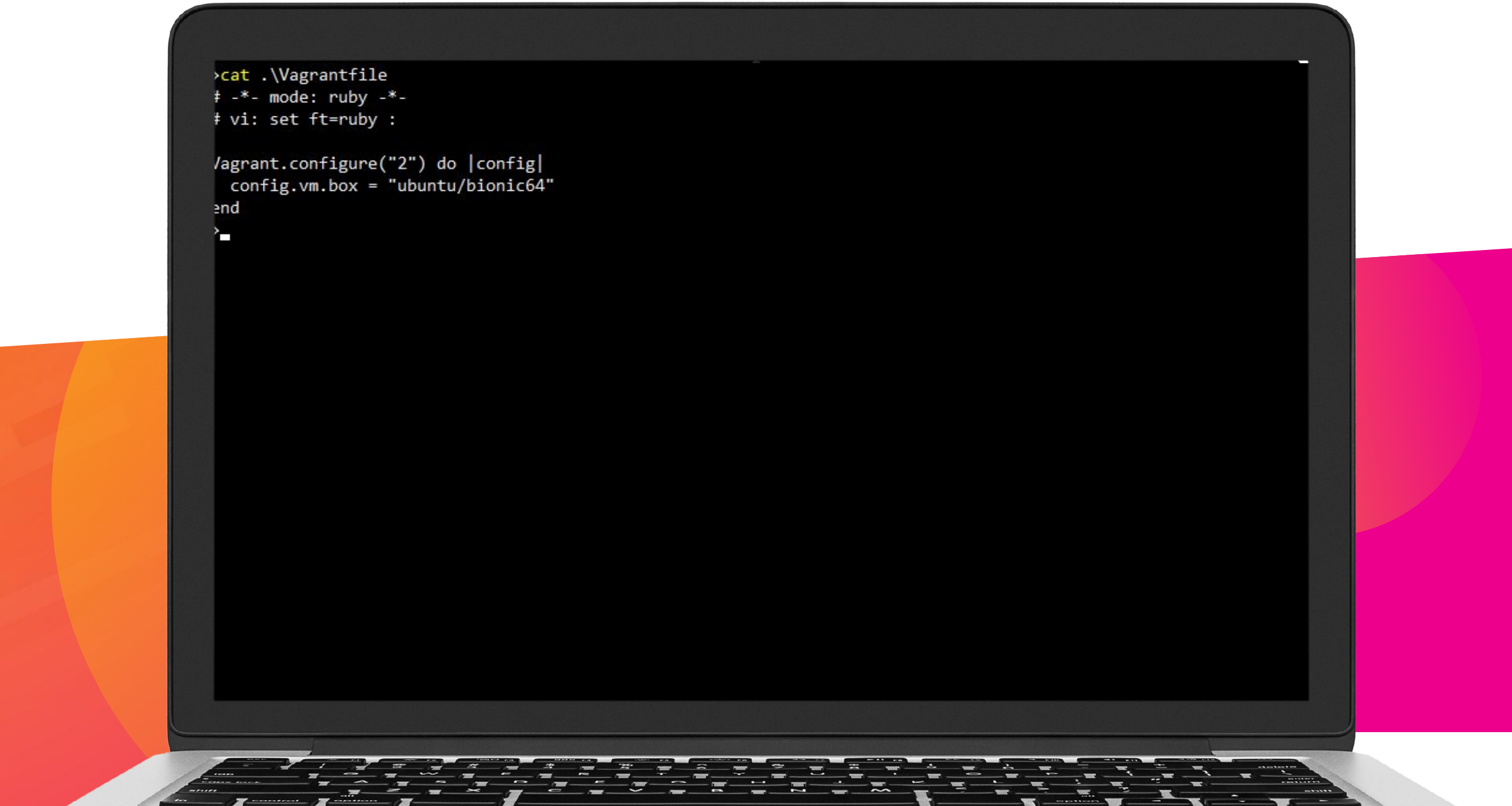
Daily

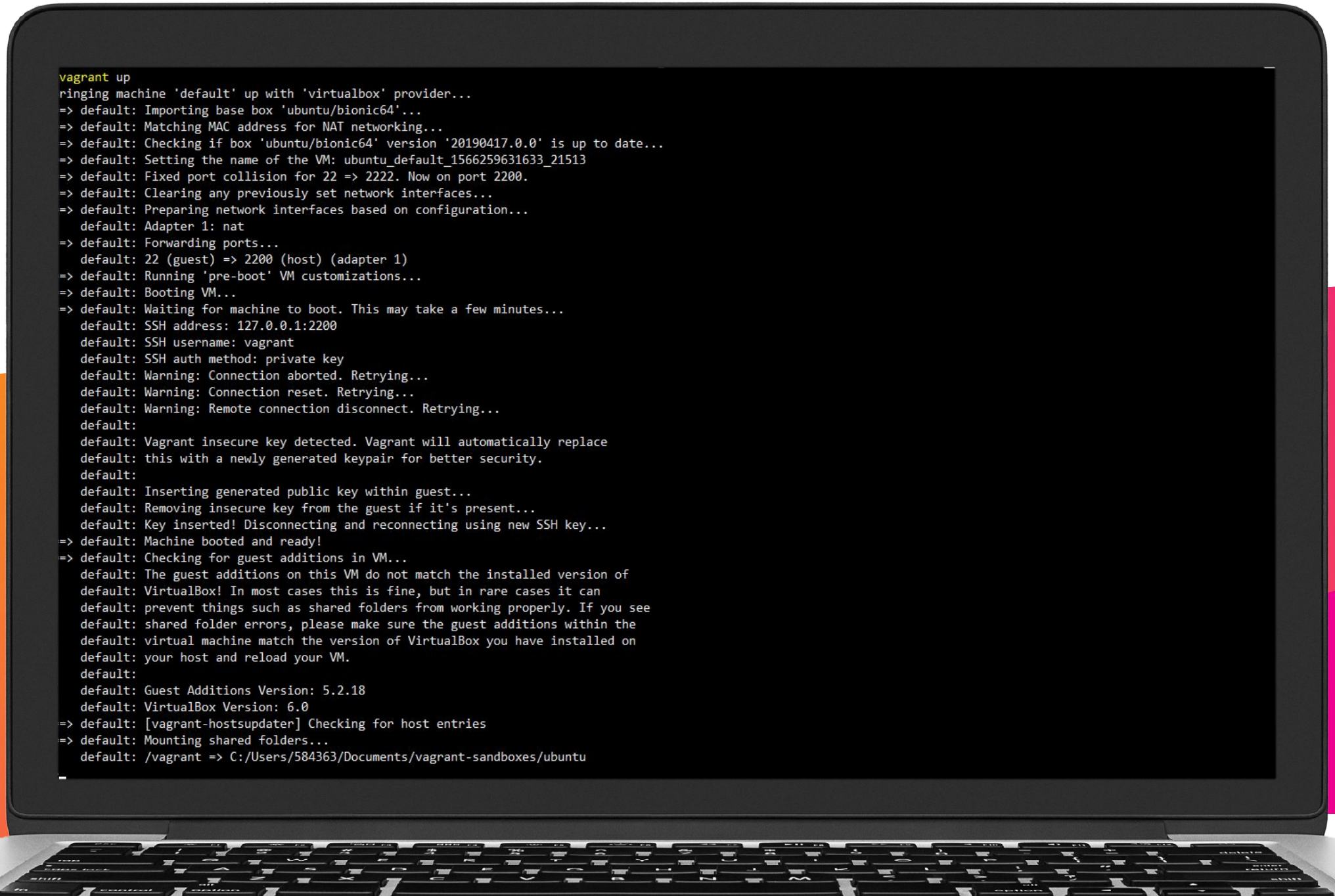
- Pull the latest commits from the repo
- Run Vagrant
- Start making code changes and see them reflected in Splunk

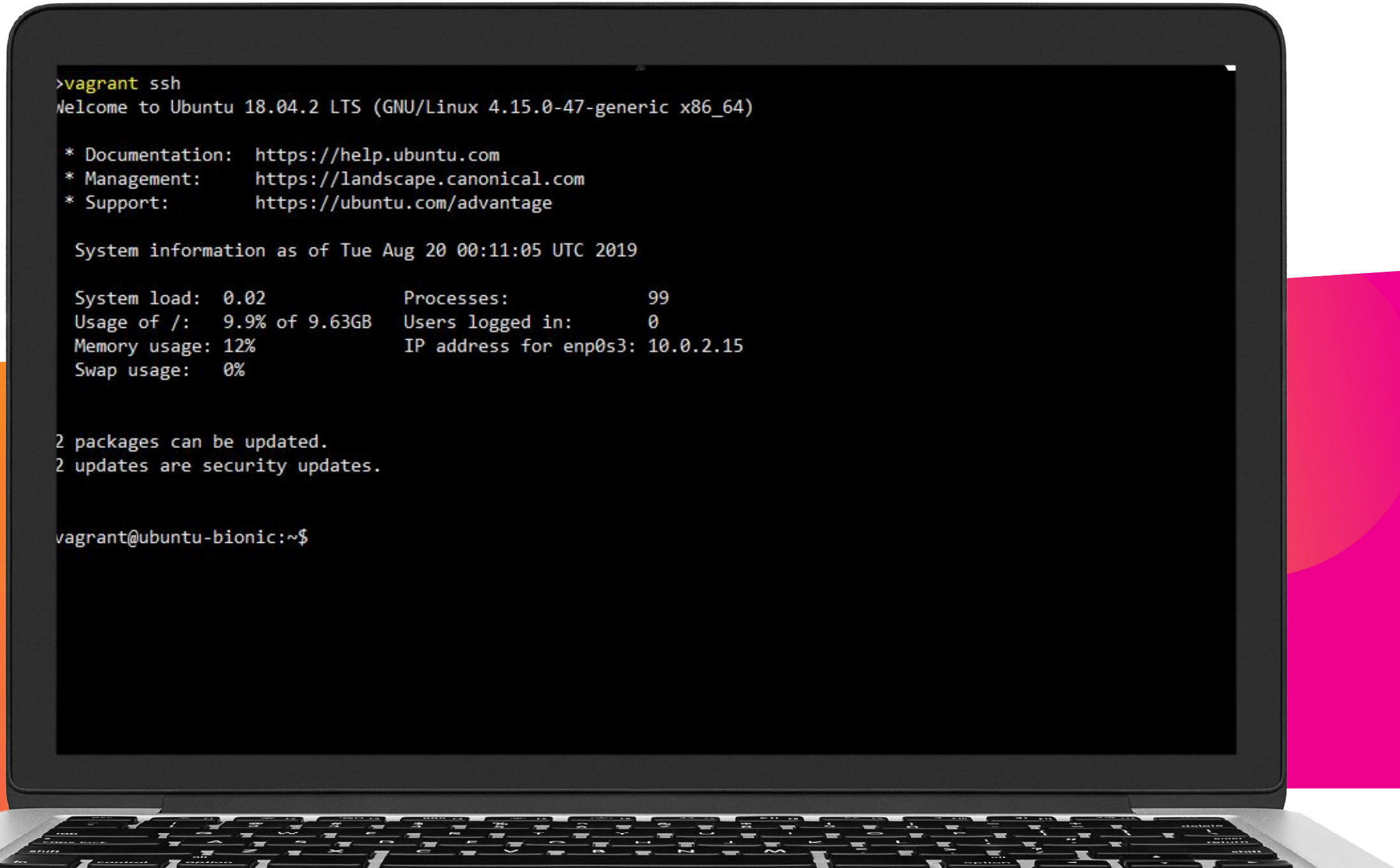
Demo

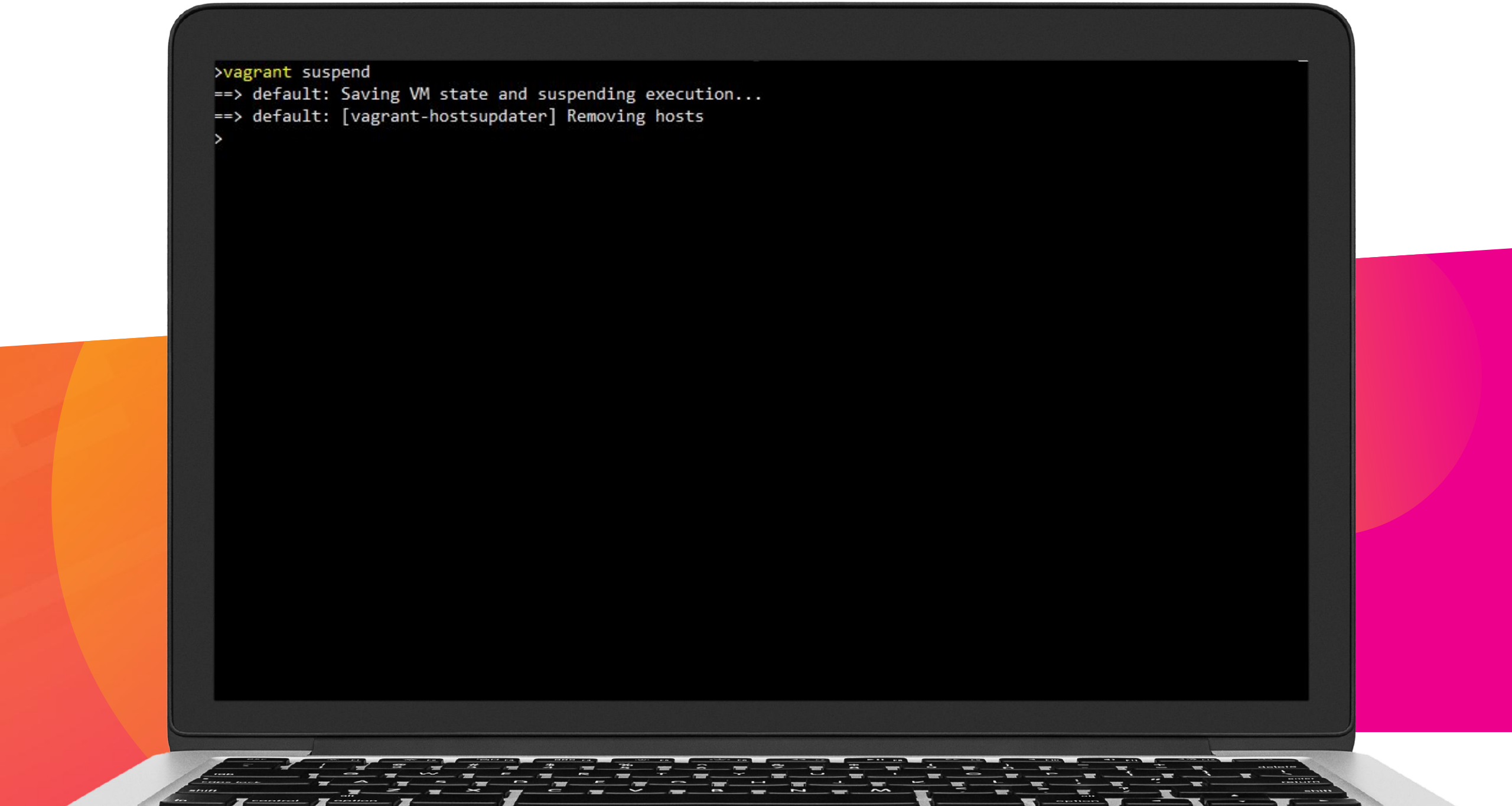


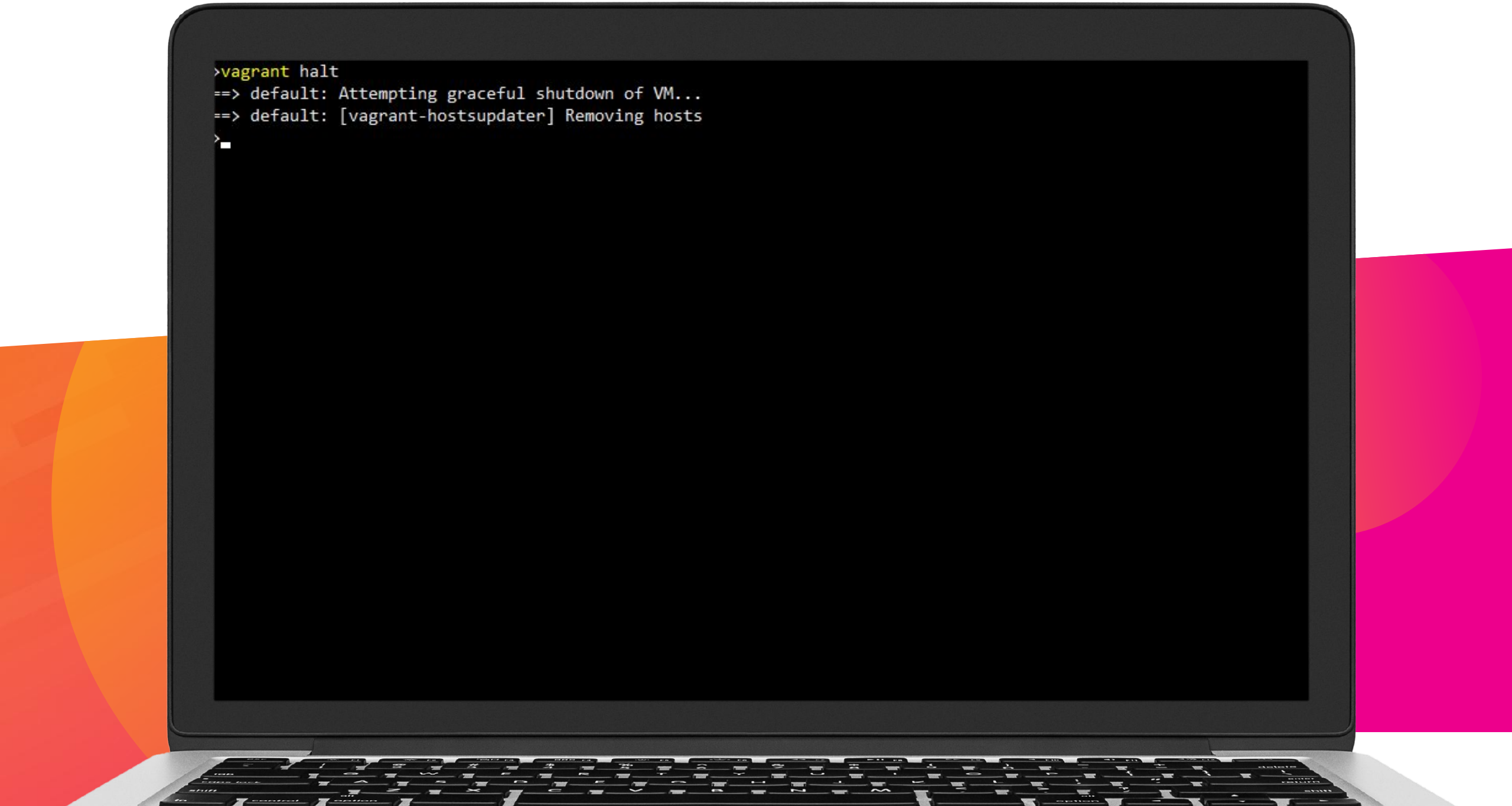


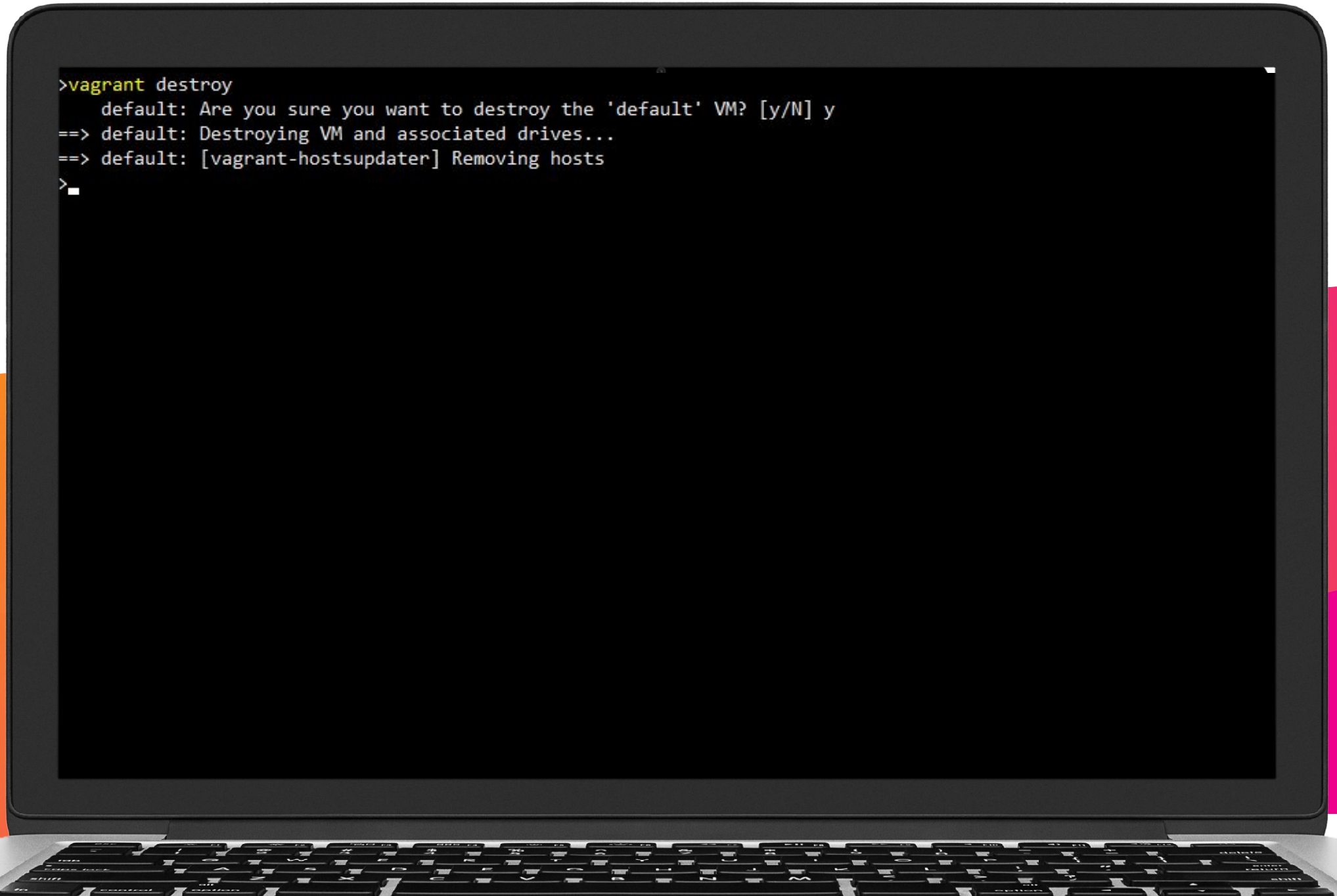


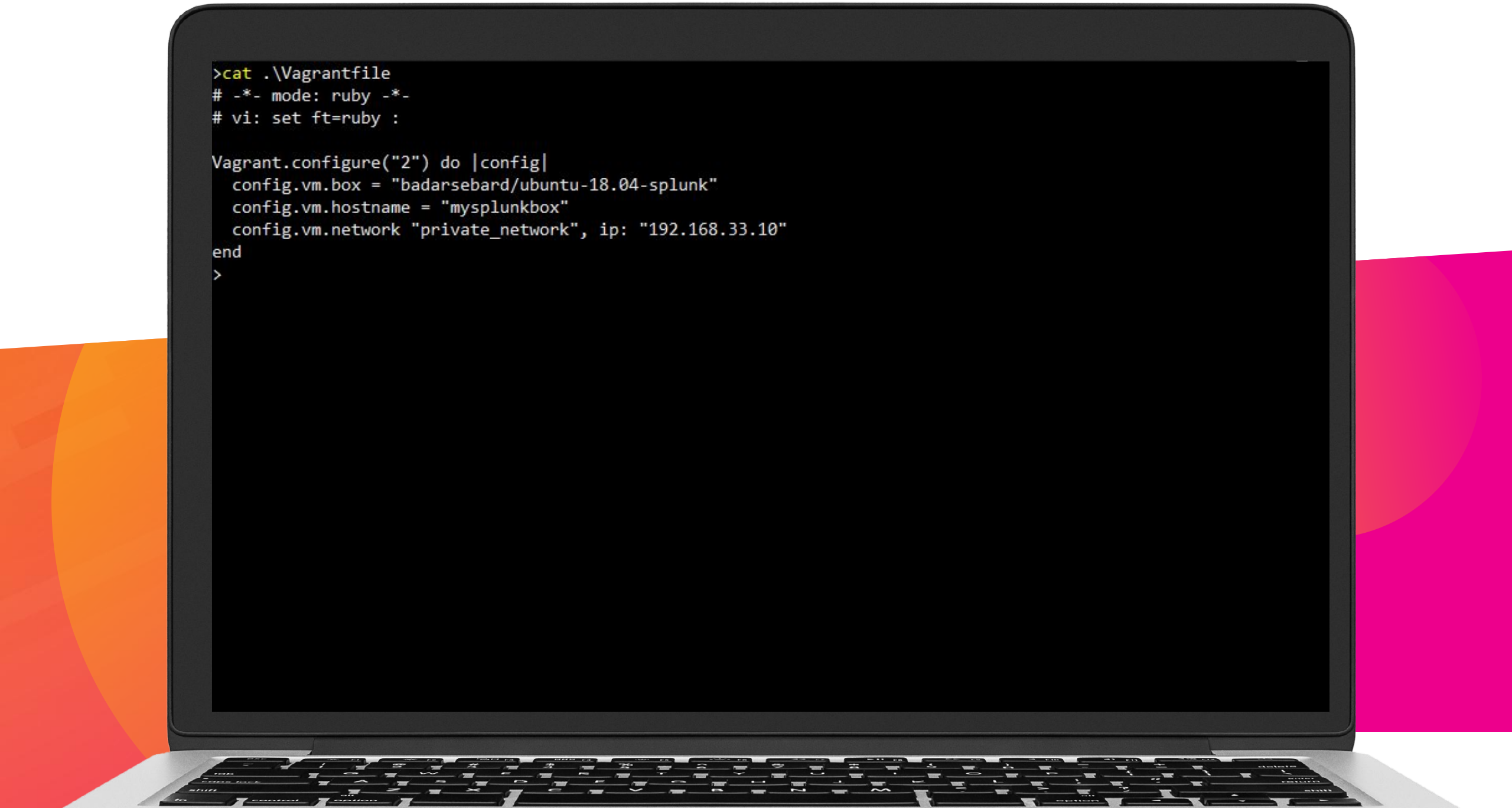


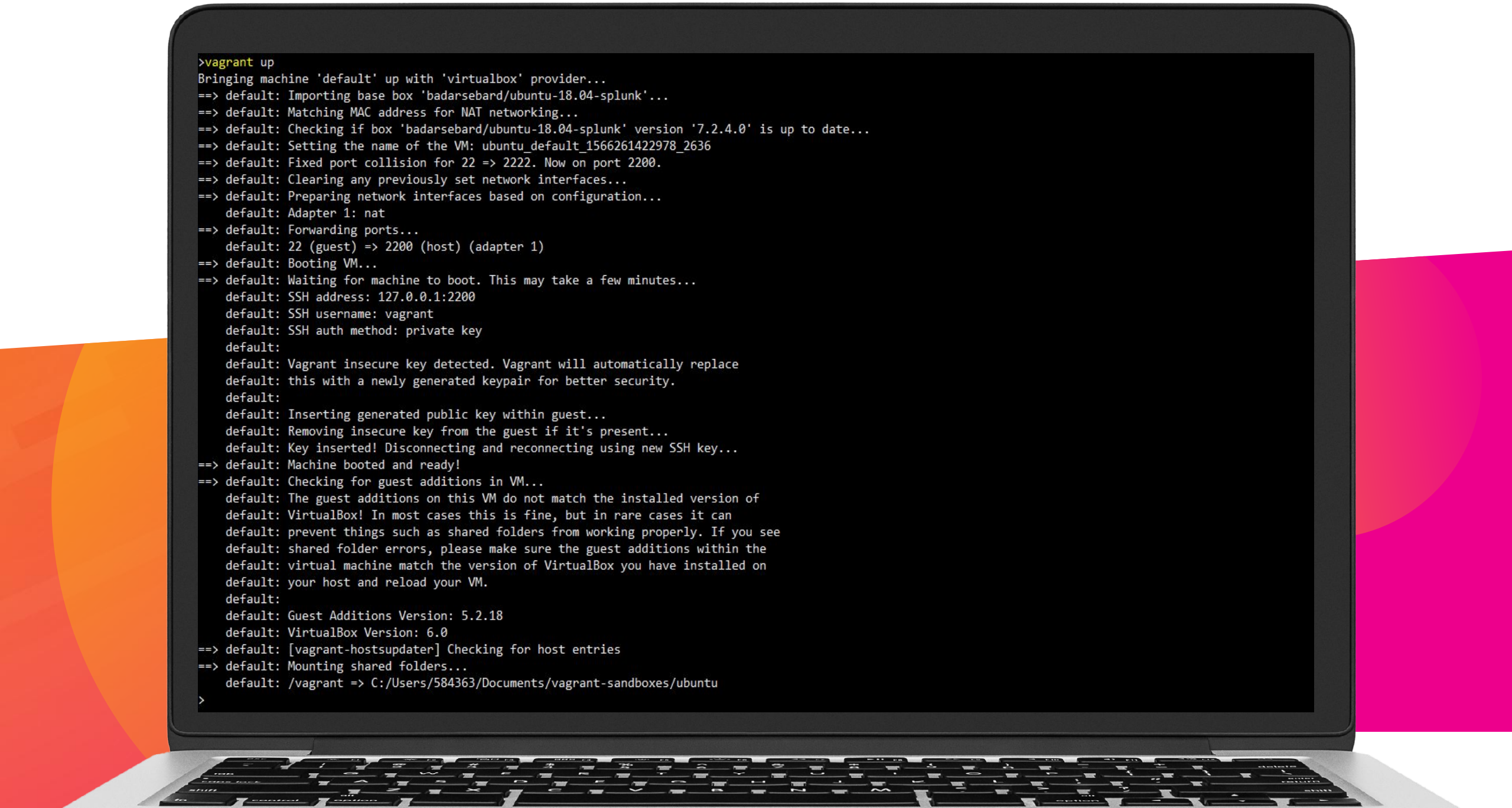


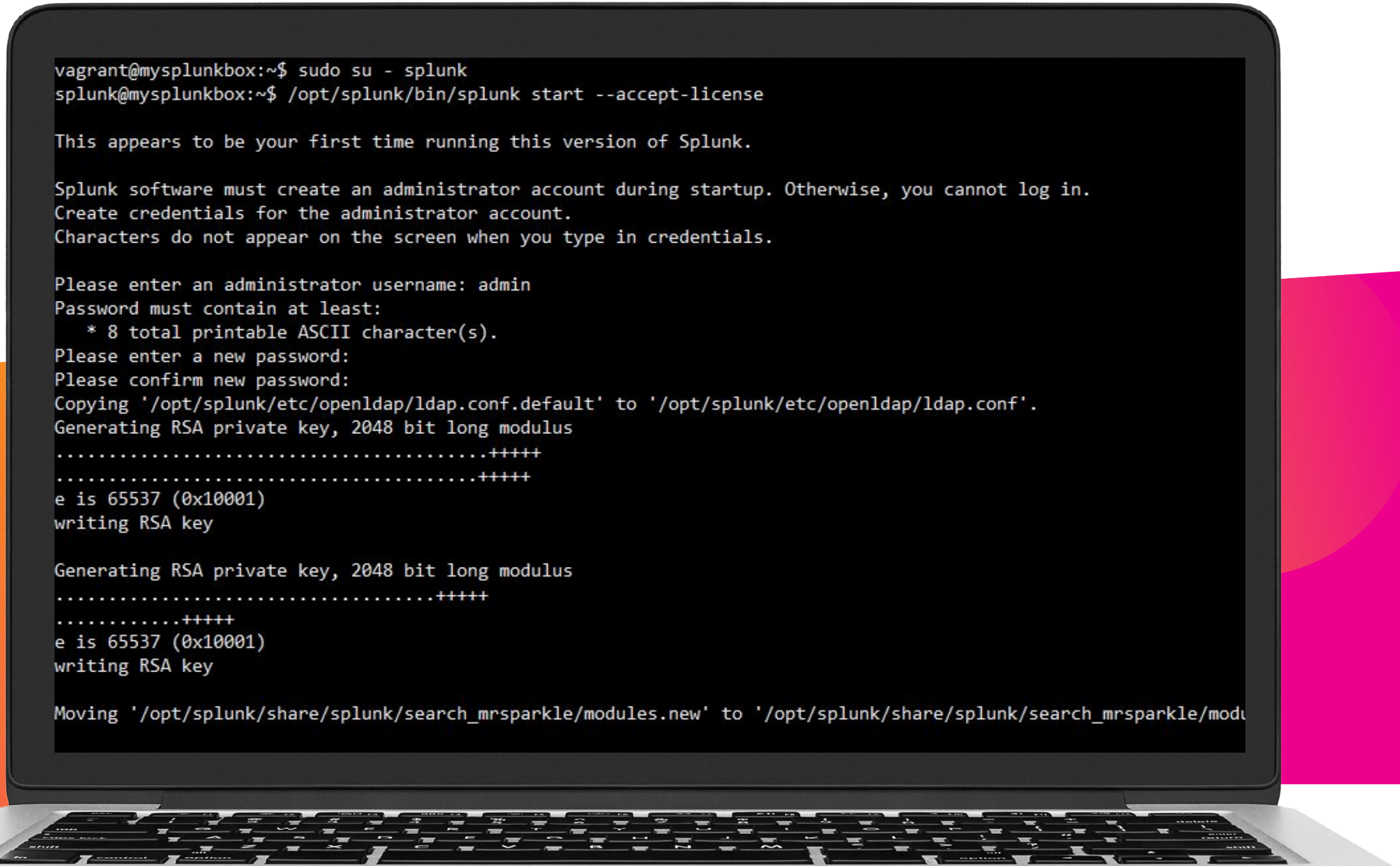


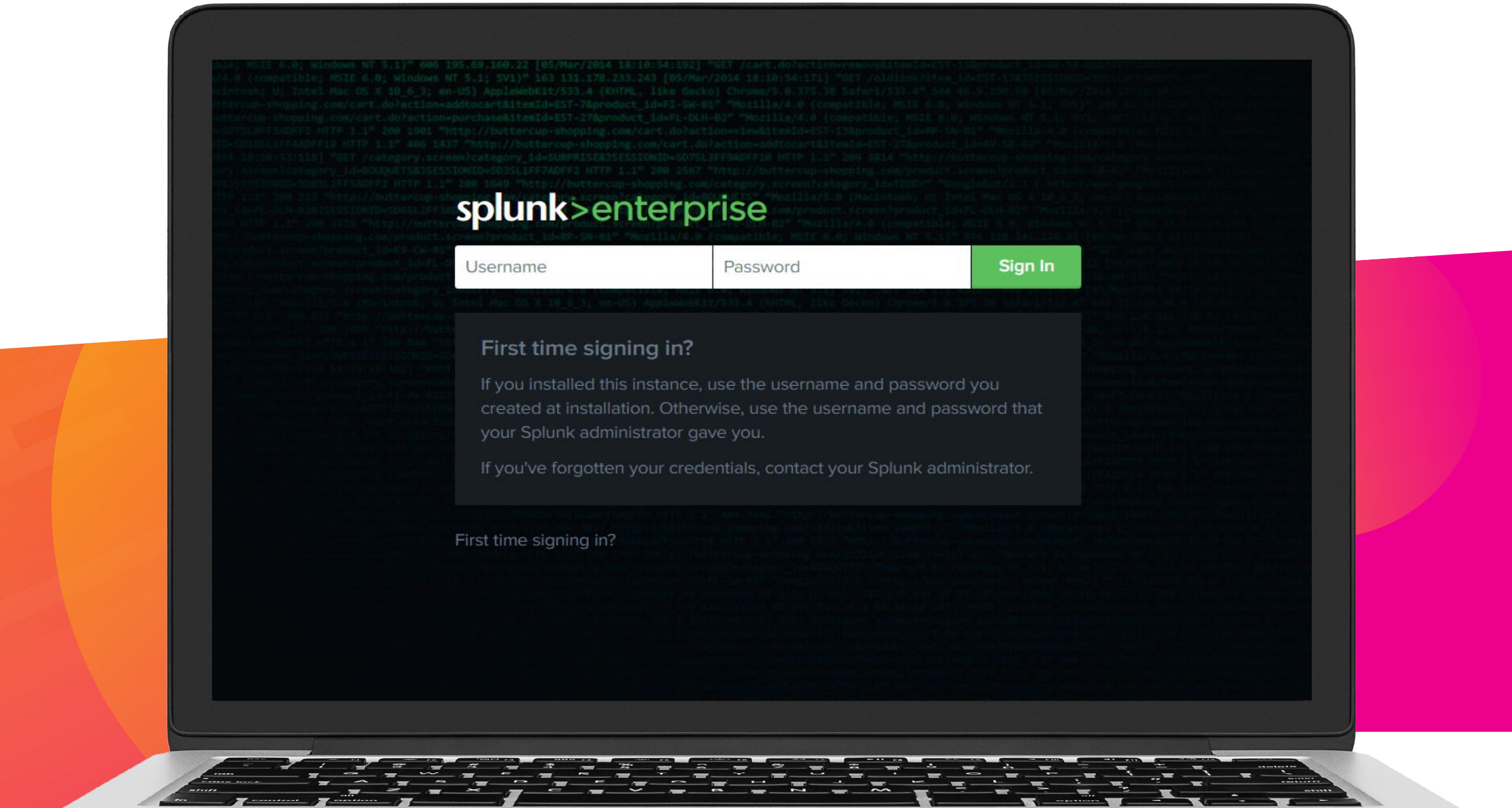












- WARNING
- Do not use Vagrant in a production environment.
- Vagrant is inherently insecure.

Case Study: PyDen

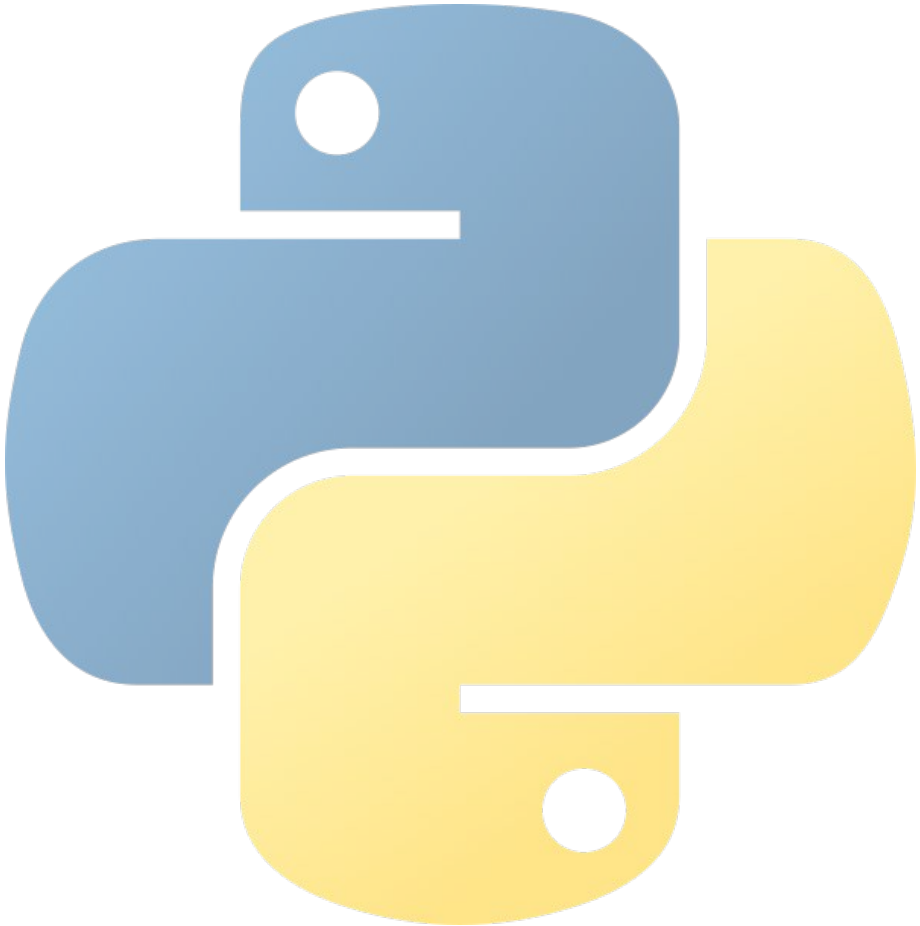
The start of my journey with Vagrant

splunk> .conf19

PyDen

Why have just one Python?

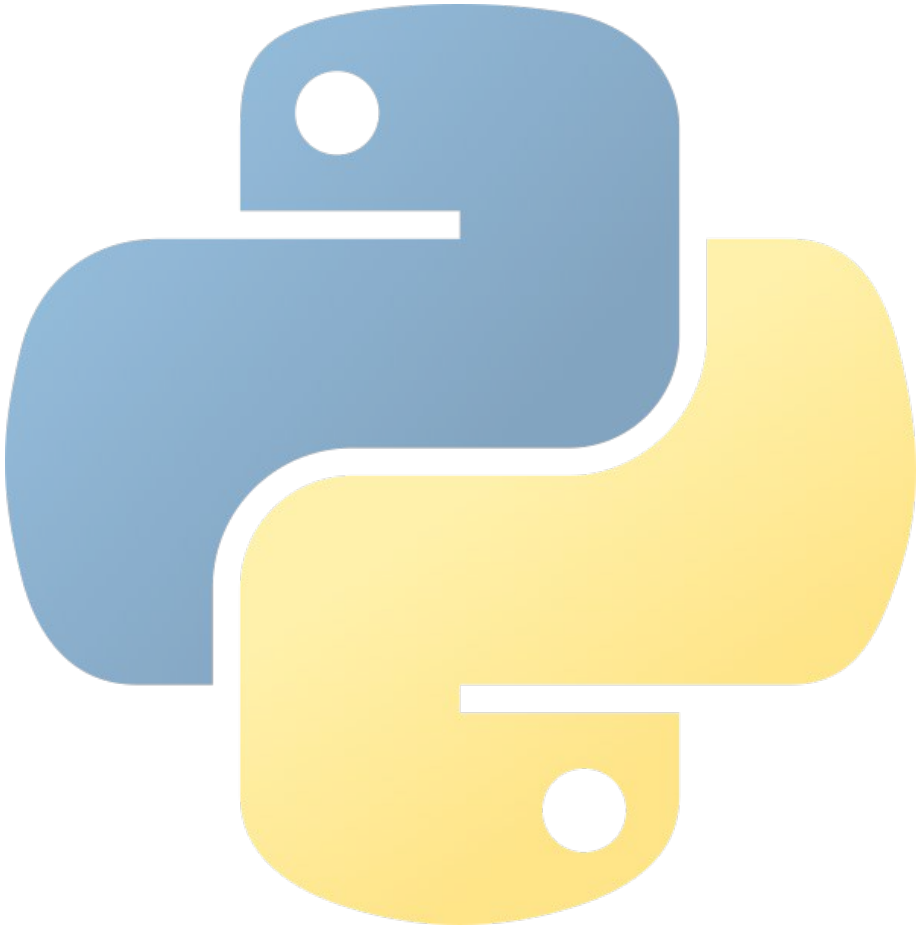
- PyDen is an app for leveraging multiple Python versions, virtual environments, and the Python Package Index inside Splunk
- Deploying the app is a two step process:
  - Create the Python distributions and virtual environments on a staging server (e.g. deployer)
  - Deploy the prepared app out to the production servers (e.g. search heads, indexers)
- Developed using Vagrant with multiple architectures
- https://github.com/badarsebard/pyden-suite



PyDen

The behind the scenes tale

- Builds Python from source
- Does a process swap and the OS level.
- Multiple components
- Multiple architectures supported.



Vagabond

The Vagrant for Splunk Project

splunk> .conf19

Vagabond

A rogue. A knave. A rapscallion!

- Open-source “barebones” Splunk project utilizing Vagrant.
- Hosted on GitLab and GitHub.
  - https://gitlab.com/badarsebard/vagabond
  - https://github.com/badarsebard/vagabond
- Jumping off point for Splunk projects.
- Supports developing multiple apps within single project
- Examples of multi-machine architectures
- Includes (optional) reference to a Splunk base box built on Ubuntu 18

Q&A

splunk> .conf19

Contact

Email: jason.rauen@gmail.com
Slack: Badarsebard
Linkedin: www.linkedin.com/in/jason-rauen

.conf19

splunk>

Thank You !

Go to the .conf19 mobile app to

RATE THIS SESSION