

Extracted from:

# Deploying Rails

Automate, Deploy, Scale, Maintain, and Sleep at Night

This PDF file contains pages extracted from *Deploying Rails*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printer versions; the content is otherwise identical.

Copyright © 2010 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The  
Pragmatic  
Programmers

# Deploying Rails

Automate, Deploy,  
Scale, Maintain,  
and Sleep at Night



Anthony Burns  
and Tom Copeland

*Edited by Brian P. Hogan*

# Deploying Rails

Automate, Deploy, Scale, Maintain, and Sleep at Night

Anthony Burns

Tom Copeland

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at <http://pragprog.com>.

Copyright © 2012 Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Printed in the United States of America.

ISBN-13: 978-1-93435-695-1

Printed on acid-free paper.

Book version: B1.0—January 19, 2012

## 2.3 Running Multiple VMs

Vagrant enables us to run multiple guest VMs on a single host. This is handy for all sorts of things and we'll use this feature extensively throughout our exercises. Let's give it a try now; first we need a new directory to hold our Vagrantfile:

```
$ mkdir ~/deployingrails/multiple_vms
$ cd ~/deployingrails/multiple_vms
```

Now we'll need a Vagrantfile. The syntax to define two separate configurations within our main configuration is to call the `define` method for each with a different name:

```
Vagrant::Config.run do |config|
  config.vm.define :app do |app_config|
  end
  config.vm.define :db do |db_config|
  end
end
```

We'll want to set `vm.name` and a memory size for each VM:

```
Vagrant::Config.run do |config|
  config.vm.define :app do |app_config|
    app_config.vm.customize do |vm|
      vm.name = "app"
      vm.memory_size = 512
    end
  end
  config.vm.define :db do |db_config|
    db_config.vm.customize do |vm|
      vm.name = "db"
      vm.memory_size = 512
    end
  end
end
```

We'll use the same box name for each VM, but we don't need to forward port 80 to the db VM, and we need to assign each VM a separate IP address. Let's add these settings to complete our Vagrantfile:

```
Vagrant::Config.run do |config|
  config.vm.define :app do |app_config|
    app_config.vm.customize do |vm|
      vm.name = "app"
      vm.memory_size = 512
    end
    app_config.vm.box = "lucid64_with_ruby192"
    app_config.vm.host_name = "app"
  end
end
```

```

app_config.vm.forward_port "ssh", 22, 2222, :auto => true
app_config.vm.forward_port "web", 80, 4567
app_config.vm.network "33.33.13.37"
end
config.vm.define :db do |db_config|
  db_config.vm.customize do |vm|
    vm.name = "db"
    vm.memory_size = 512
  end
  db_config.vm.box = "lucid64_with_ruby192"
  db_config.vm.host_name = "db"
  db_config.vm.forward_port "ssh", 22, 2222, :auto => true
  db_config.vm.network "33.33.13.38"
end
end

```

Our VMs are defined, so we can start them both with `vagrant up` and we'll see output as both VMs are started. Note that the output for each VM is prefixed with the VM name:

```

$ vagrant up
[app] Importing base box 'lucid64_with_ruby192'...
[app] Preparing host only network...
[app] Matching MAC address for NAT networking...
[app] Clearing any previously set forwarded ports...
[app] Forwarding ports...
[app] -- ssh: 22 => 2222 (adapter 1)
[app] -- web: 80 => 4567 (adapter 1)
[app] Creating shared folders metadata...
[app] Running any VM customizations...
[app] Booting VM...
[app] Waiting for VM to boot. This can take a few minutes.
[app] VM booted and ready for use!
[app] Enabling host only network...
[app] Setting host name...
[app] Mounting shared folders...
[app] -- v-root: /vagrant
[db] Fixed port collision 'ssh'. Now on port 2200.
[db] Importing base box 'lucid64_with_ruby192'...
[db] Preparing host only network...
[db] Matching MAC address for NAT networking...
[db] Clearing any previously set forwarded ports...
[db] Forwarding ports...
[db] -- ssh: 22 => 2200 (adapter 1)
[db] Creating shared folders metadata...
[db] Running any VM customizations...
[db] Booting VM...
[db] Waiting for VM to boot. This can take a few minutes.
[db] VM booted and ready for use!
[db] Enabling host only network...

```

```
[db] Setting host name...
[db] Mounting shared folders...
[db] -- v-root: /vagrant
```

We can connect into each VM using our usual vagrant ssh, but this time we'll also specify the VM name:

```
$ vagrant ssh app
Last login: Wed Dec 21 19:47:36 2011 from 10.0.2.2
vagrant@app:~$ hostname
app
vagrant@app:~$ exit
logout
$ vagrant ssh db
Last login: Thu Dec 22 21:19:54 2011 from 10.0.2.2
vagrant@db:~$ hostname
db
```

Generally, when we want to run any Vagrant command on one VM in a multiple VM set up we need to specify the VM name. For some commands (e.g., vagrant halt) this is optional and we can act on both VMs by not specifying a host name.

We can connect from the db VM over to the app VM via ssh as the vagrant user with a password of vagrant:

```
db $ ssh 33.33.13.37
The authenticity of host '33.33.13.37 (33.33.13.37)' can't be established.
RSA key fingerprint is ed:d8:51:8c:ed:37:b3:37:2a:0f:28:1f:2f:1a:52:8a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '33.33.13.37' (RSA) to the list of known hosts.
vagrant@33.33.13.37's password:
Last login: Thu Dec 22 21:19:41 2011 from 10.0.2.2
vagrant@app:~$
```

Finally, we can shut down and destroy both VMs with vagrant destroy:

```
$ vagrant destroy
[app] Forcing shutdown of VM...
[app] Destroying VM and associated drives...
[db] Forcing shutdown of VM...
[db] Destroying VM and associated drives...
```

We're using two VMs here, but Vagrant can handle as many VMs as you need up to the resource limits of your computer. So that ten node Hadoop cluster is finally a reality.