Install log for: ruby

=> Downloading ...

=> Extracting archive...

=> Compiling...

=> Installing...

=> Activating...

=> Installed ruby 2.0.0

============ ruby ============

Ruby and rbenv (Ruby Version Manager) are preinstalled on all Codio Boxes.

http://flowingdata.com/2012/08/02/how-to-make-an-interactive-network-visualization/

Install log for: apache2

=> Downloading http://parts.codio.com/box-codio-v1/apr-1.5.1-binary.tar.gz...

=> Extracting archive...

=> Installing...

=> Activating...

=> Installed apr 1.5.1

=> Downloading http://parts.codio.com/box-codio-v1/apr\_util-1.5.3-binary.tar.gz...

=> Extracting archive...

=> Installing...

=> Activating...

=> Installed apr\_util 1.5.3

=> Downloading http://parts.codio.com/box-codio-v1/apache2-2.4.9-binary.tar.gz...

=> Extracting archive...

=> Installing...

=> Activating...

=> Installed apache2 2.4.9

============ apache2 ============

To start the Apache server:

$ parts start apache2

To stop the Apache server:

$ parts stop apache2

Apache config is located at:

$ /home/codio/.parts/etc/apache2/httpd.conf

Default document root is located at:

$ /home/codio/workspace

git remote add origin git@github.com:johnfkraus/aqlist-visualization.git

git push -u origin master

With define you register a module in require.js that you than can depend on in other module definitions or

require statements. With require you "just" load/use a module or javascript file that can be loaded by require.js.

For examples have a look at the documentation

My rule of thumb:

Define: If you want to declare a module other parts of your application will depend on.

Require: If you just want to load and use stuff.

…or create a new repository on the command line

GIT

====

touch README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin git@github.com:johnfkraus/aq-list-viz.git

git push -u origin master

…or push an existing repository from the command line

git remote add origin git@github.com:johnfkraus/aq-list-viz.git

git push -u origin master

…or import code from another repository

overwrite remote:

git push -f origin master

SSH / Generating SSH keys

Generating SSH keys

SSH keys are a way to identify trusted computers, without involving passwords.

The steps below will walk you through generating an SSH key and then adding the public key to your GitHub account.

Tip: We recommend that you regularly review your SSH keys list and revoke any that haven't been used in a while.

Step 1: Check for SSH keys

First, we need to check for existing SSH keys on your computer. Open up your Git Bash and type:

ls -al ~/.ssh

# Lists the files in your .ssh directory, if they exist

Check the directory listing to see if you already have a public SSH key. The default public key file names are:

id\_dsa.pub

id\_ecdsa.pub

id\_ed25519.pub

id\_rsa.pub

Step 2: Generate a new SSH key

To generate a new SSH key, copy and paste the text below, making sure to substitute in your email address.

The default settings are preferred, so when you're prompted to "Enter a file in which to save the key",

just press Enter to continue.

ssh-keygen -t rsa -C "your\_email@example.com"

ssh-keygen -t rsa -C "johnkraus3@gmail.com"

# Creates a new ssh key, using the provided email as a label

# Generating public/private rsa key pair.

# Enter file in which to save the key (/c/Users/you/.ssh/id\_rsa): [Press enter]

Next, you'll be asked to enter a passphrase.

Tip: We strongly recommend a very good, secure passphrase. For more information, see Working with SSH key passphrases.

# Enter passphrase (empty for no passphrase): [Type a passphrase]

# Enter same passphrase again: [Type passphrase again]

Which should give you something like this:

# Your identification has been saved in /c/Users/you/.ssh/id\_rsa.

# Your public key has been saved in /c/Users/you/.ssh/id\_rsa.pub.

# The key fingerprint is:

# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your\_email@example.com

Then add your new key to the ssh-agent:

USE GIT BASH

# start the ssh-agent in the background

ssh-agent -s

# Agent pid 59566

ssh-add ~/.ssh/id\_rsa

ssh-add .ssh/id\_rsa

$ ssh-add .ssh/id\_rsa

Could not open a connection to your authentication agent.

User1@LENOVO-PC ~

$

eval `ssh-agent -s`

Agent pid 3556

User1@LENOVO-PC ~

$ ssh-add

Enter passphrase for /c/Users/User1/.ssh/id\_rsa:

Identity added: /c/Users/User1/.ssh/id\_rsa (/c/Users/User1/.ssh/id\_rsa)

Step 3: Add your SSH key to your account

Run the following command to copy the key to your clipboard. Keep in mind that your key may also be

named id\_dsa.pub, id\_ecdsa.pub or id\_ed25519.pub.

clip < ~/.ssh/id\_rsa.pub

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDL+32kJRdWgDQBzSXbVakur0lnup3wUQXn5OL0hCDNLkTsHRVtvFCvopOaFCYr0EGWDuhNKOlR0oWFTkZI1jJrrvAMtz7VXhS+OR5tntEB7EIiug2miU3b6Kg2OojoChuk2iW28BWjGrlZhvSCeUxE8PSj+5szSPSCdyWkRsRw47WnS8rqZzijumIaDFB5qaPsVjWCWRuKrlWQVoS5BA/597rXNjcd9XfAcEp1znDglM+SBsxE/sfu32mYpGudjDfav4QxfzWAp6P8sqF4fJJ5CeKtUWx8Xe11saW32xbn8B+lgpzg7jcF1eV9VJiBvt53YckLJte7MjAp/L5/BaNX johnkraus3@gmail.com

# Copies the contents of the id\_rsa.pub file to your clipboard

Alternatively, using your favorite text editor, you can open the public key file and copy the contents of the file manually.

Now that you have the key copied, it's time to add it to GitHub:

Settings icon in the user bar

In the top right corner of any page, click .

SSH keysIn the user settings sidebar, click SSH keys.

SSH Key button

Click Add SSH key.

In the Title field, add a descriptive label for the new key. For example, if you're using a personal Mac, you might call this key "Personal MacBook Air".

The key fieldPaste your key into the "Key" field.

The Add key buttonClick Add key.

Confirm the action by entering your GitHub password.

Step 4: Test everything out

To make sure everything is working, you'll now try SSHing to GitHub. When you do this, you will be asked to authenticate this action using your password, which was the passphrase you created earlier.

Open up your Git Bash and type:

ssh -T git@github.com

# Attempts to ssh to GitHub

You may see this warning:

# The authenticity of host 'github.com (207.97.227.239)' can't be established.

# RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.

# Are you sure you want to continue connecting (yes/no)?

Don't worry! This is supposed to happen. Verify that the fingerprint in your terminal matches the one we've provided up above, and then type "yes."

# Hi username! You've successfully authenticated, but GitHub does not

# provide shell access.

If that username is yours, you've successfully set up your SSH key! Don't worry about the "shell access" thing, you don't want that anyway.

If you receive a message about "access denied," you can read these instructions for diagnosing the issue.

If you're switching from HTTPS to SSH, you'll now need to update your remote repository URLs. For more information, see Changing a remote's URL.

HEROKU ERROR

Argument for @NotNull parameter 'e' of com/intellij/openapi/ActionSystem/AnAction.update must not be null

Short path to browser:

"C:\Program Files\Internet Explorer\iexplore.exe"

cmd /c for %A in ("C:\Documents and Settings\User\NTUSER.DAT") do @echo %~sA

cmd /c for %A in ("C:\Program Files\Internet Explorer\iexplore.exe") do @echo %~sA

C:\PROGRA~1\INTERN~1\iexplore.exe

HEROKU

heroku login

git clone https://github.com/heroku/node-js-getting-started.git

cd node-js-getting-started/

heroku create --http-git

git push heroku master

heroku ps:scale web=1

heroku open

heroku logs --tail

cd .ssh

clip < ~/.ssh/id\_rsa.pub

heroku auth:token // display the authorization token; also found in ~/\_netrc

curl -H "Accept: application/json" -n https://api.heroku.com/apps

heroku keys:add

heroku keys:remove adam@workstation.local

heroku keys

Authenticating with the API token

Having logged in, you can use curl to access the Heroku API:

$ curl -H "Accept: application/json" -n https://api.heroku.com/apps

You can also create a file ~/.curlrc, containing extra command-line options for curl:

~/.curlrc

--netrc

--header "Accept: application/json"

With this file, the command is simply:

$ curl https://api.heroku.com/apps

File format

The file contains a list of free-form records and comments. Comments start with a # (hash) symbol and continue

to the end of the line. Each record is of the form:

machine api.heroku.com

login me@example.com

password ABC123

One other type of record, macdef, can appear in .netrc files, but it is not commonly used and is ignored by the heroku command.

Keep reading

Validate the connection

========================

You can check to see whether your keys are working by trying the following command:

ssh -v git@heroku.com

Data structures, set and bag were from

https://www.npmjs.org/package/backpack-node

HowTo: Setup SSH Keys on a Linux / Unix System

by NIXCRAFT on MARCH 9, 2014 · 11 COMMENTS· LAST UPDATED OCTOBER 24, 2014

in CENTOS, CRYPTOGRAPHY, DEBIAN / UBUNTU, LINUX, OPENBSD, REDHAT AND FRIENDS, UNIX

I recently read that SSH keys provide a secure way of logging into a Linux and Unix-based server. How do I set up SSH keys on a Linux or Unix based systems? In SSH for Linux/Unix, how do I set up public key authentication?

Tutorial details

Difficulty Easy (rss)

Root privileges No

Requirements OpenSSH client and server

I am assuming that you are using Linux or Unix-like server and client with the following software:

OpenSSH SSHD server

OpenSSH ssh client and friends on Linux (Ubuntu, Debian, {Free,Open,Net}BSD, RHEL, CentOS, OSX and co).

What is a public key authentication?

OpenSSH server supports various authentication schema. The two most popular are as follows:

Passwords based authentication

Public key based authentication. It is an alternative security method to using passwords. This method is recommended on a VPS, cloud, dedicated or even home based server.

How do I set up public key authentication?

You must generate both a public and a private key pair. For example:

///////////

//Internet//

////////////

|

+---------------+ | +-------------+

| Unix/Linux | | | Linux/Unix |

| Server with +------+-------+ OSX/\*BSD |

| OpenSSH SSHD | | Client |

+---------------+ +-------------+

server1.cyberciti.biz client1.cyberciti.biz

75.126.153.206 192.168.1.42

Where,

server1.cyberciti.biz - You store your public key on the remote hosts and you have an accounts on this Linux/Unix based server.

client1.cyberciti.biz - Your private key stays on the desktop/laptop/ computer (or local server) you use to connect to server1.cyberciti.biz server. Do not share or give your private file to anyone.

In public key based method you can log into remote hosts and server, and transfer files to them, without using your account passwords. Feel free to replace server1.cyberciti.biz and client1.cyberciti.biz names with your actual setup. Enough talk, let's set up public key authentication. Open the Terminal and type following commands if .ssh directory does not exists:

mkdir -p $HOME/.ssh

chmod 0700 $HOME/.ssh

#1: Create the key pair

On the computer (such as client1.cyberciti.biz), generate a key pair for the protocol.

ssh-keygen -t rsa

Sample outputs:

Generating public/private rsa key pair.

Enter file in which to save the key (/Users/vivek/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /Users/vivek/.ssh/id\_rsa.

Your public key has been saved in /Users/vivek/.ssh/id\_rsa.pub.

The key fingerprint is:

80:5f:25:7c:f4:90:aa:e1:f4:a0:01:43:4e:e8:bc:f5 vivek@desktop01

The key's randomart image is:

+--[ RSA 2048]----+

| oo ...+. |

|.oo . .ooo |

|o .o. . .o . |

| o ...+o. |

| o .=.=S |

| . .Eo . |

| |

| |

| |

+-----------------+

You need to set the Key Pair location and name. I recommend you use the default location if you do not yet have another key there, for example: $HOME/.ssh/id\_rsa. You will be prompted to supply a passphrase (password) for your private key. I suggest that you setup a passphrase when prompted. You should see two new files in $HOME/.ssh/ directory:

$HOME/.ssh/id\_rsa - contains your private key.

$HOME/.ssh/id\_rsa.pub - contain your public key.

#2: Install the public key in remote server

Use scp or ssh-copy-id command to copy your public key file (e.g., $HOME/.ssh/id\_rsa.pub) to your account on the remote server/host (e.g., nixcraft@server1.cyberciti.biz). To do so, enter the following command on your client1.cyberciti.biz:

ssh-copy-id -i $HOME/.ssh/id\_rsa.pub user@server1.cyberciti.biz

OR just copy the public key in remote server as authorized\_keys in ~/.ssh/ directory:

scp $HOME/.ssh/id\_rsa.pub user@server1.cyberciti.biz:~/.ssh/authorized\_keys

A note about appending the public key in remote server

On some system ssh-copy-id command may not be installed, so use the following commands (when prompted provide the password for remote user account called vivek) to install and append the public key:

## First create .ssh directory on server ##

ssh vivek@server1.cyberciti.biz umask 077; test -d .ssh || mkdir .ssh

## cat local id.rsa.pub file and pipe over ssh to append the public key in remote server ##

cat $HOME/.ssh/id\_rsa.pub | ssh vivek@server1.cyberciti.biz cat >> .ssh/authorized\_keys

#3: Test it (type command on client1.cyberciti.biz)

The syntax is:

ssh user@server1.cyberciti.biz

Or copy a text file called foo.txt:

scp foo.txt user@server1.cyberciti.biz:/tmp/

You will be prompted for a passphrase. To get rid of passphrase whenever you log in the remote host, try ssh-agent and ssh-add commands.

What are ssh-agent and ssh-add, and how do I use them?

To get rid of a passphrase for the current session, add a passphrase to ssh-agent and you will not be prompted for it when using ssh or scp/sftp/rsync to connect to hosts with your public key. The syntax is as follows:

>>>>>>>>>>>>>>>>

eval $(ssh-agent)

>>>>>>>>>>>>

Type the ssh-add command to prompt the user for a private key passphrase and adds it to the list maintained by ssh-agent command:

ssh-add

Enter your private key passphrase. Now try again to log into user@server1.cyberciti.biz and you will not be prompted for a password:

ssh user@server1.cyberciti.biz

#4: Disable the password based login on a server

Login to your server, type:

## client commands ##

eval $(ssh-agent)

ssh-add

ssh user@server1.cyberciti.biz<

Edit /etc/ssh/sshd\_config on server1.cyberciti.biz using a text editor such as nano or vim:

$ sudo vim /etc/ssh/sshd\_config

OR directly jump to PermitRootLogin line using a vim text editor:

$ sudo vim +/PermitRootLogin /etc/ssh/sshd\_config

Find PermitRootLogin and set it as follows:

PermitRootLogin no

Save and close the file.

Reload/restart sshd server, type command as per your Linux/Unix version:

## CentOS/RHEL/Fedora Linux server reload sshd ##

sudo service sshd reload

#5: How to add or replace a passphrase for an existing private key?

To to change your passphrase type the following command:

ssh-keygen -p

#6: How to backup an existing private/public key?

Just copy files to your backup server or external USB pen/hard drive:

## Copy files to home based nas server ##

rsync -avr $HOME/.ssh user@home.nas-server:/path/to/encrpted/nas/partition/

## Copy files to usb pen drive mounted at /mnt/usb ##

cp -avr $HOME/.ssh/ /mnt/usb/backups/

SEE ALSO

keychain: Set Up Secure Passwordless SSH Access For Backup Scripts

Man pages - ssh(1), ssh-agent(1), ssh-add(1), ssh-keygen(1)

To add to the story, I do this often with keys setup from my office desktop.

for i in server1 server2 server3 server4; do ssh mylogin@$i "hostname" ; done

This one line will login to the four servers and run the command hostname .

Replace hostname with your hearts desire.

–Terry

REPLY

I have my ssh keys setup with up on one server and connecting with two other servers successfully. I now have to rename and reip all three servers, i noticed with the pub key the server name is listed. When i rename and re-ip my servers is there any way to update the keys or do i have re-create again?

REPLY

7 rajesh September 23, 2014 at 4:55 pm

i have generate ssh key on A server for communicate server B. then i can able to do ssh with out passwd to server B.

but when i try to login in Server B . ..then trying to do ssh to server A ..it is asking password..why lit happens ?

it should be vise versa ..rt ? server A – server B , Server B- server A

Thanks

REPLY

A to B password less

server A:

ssh-keygen -t rsa

press enter,enter

two keys is create the path /root/.ssh

next go to the cd .ssh

ls -lrt

scp id\_pas.pub root@serverB:/home

server B:

check the cd /home

ls

cd /root/.ssh

in server B is not .ssh directory

create the directory

mkdir /root/.ssh

chmod 700 /root/.ssh

cd /root/.ssh

cp /home/id\_pas.pub authorized\_keys

already authorized\_keys in another keys

so append data because over read the data

cat>>authorized\_keys</home/id\_pas.pub -this command is append the data not over read the old data

next go to server A

ssh root@serverB

login the passwd lesss

REPLY

9 milosz October 23, 2014 at 12:33 pm

I believe you forgot the pipe character:

cat $HOME/.ssh/id\_rsa.pub ssh vivek@server1.cyberciti.biz cat >> .ssh/authorized\_keys

should be:

cat $HOME/.ssh/id\_rsa.pub | ssh vivek@server1.cyberciti.biz cat >> .ssh/authorized\_keys

REPLY

How can I access this remote server from different network ?

Related Faqs

Linux / UNIX: Generate SSH KeysLinux / UNIX: Generate SSH Keys

OpenSSH: ssh-add / ssh-agent Command Set Maximum Lifetime In Seconds OpenSSH: ssh-add / ssh-agent Command Set Maximum Lifetime In Seconds

Install / Append SSH Key In A Remote Linux / UNIX Servers Authorized\_keysInstall / Append SSH Key In A Remote Linux / UNIX Servers Authorized\_keys

ssh-vulnkey(1)) Error and Solution

CentOS Linux 5/6: Change OpenSSH Port NumberCentOS Linux 5/6: Change OpenSSH Port Number

Red Hat Linux (RHEL) 5/6: Change OpenSSH Port NumberRed Hat Linux (RHEL) 5/6: Change OpenSSH Port Number

SSH: Agent Admitted Failure To Sign Using The Key Error And SolutionSSH: Agent Admitted Failure To Sign Using The Key Error And Solution

Rackspace Cloud Files: Upload Files Using Secure FTP (SFTP) Client [ sftp-cloudfs ]Rackspace Cloud Files: Upload Files Using Secure FTP (SFTP) Client [ sftp-cloudfs ]

X11 forwarding request failed on channel 0 Error and Solution X11 forwarding request failed on channel 0 Error and Solution

5 Awesome Open Source Backup Software For Linux and Unix-like Systems

============================

Modulus

http://blog.modulus.io/nodejs-and-express-static-content

JANUARY 21, 2013

CHARLIE KEY

@ZWIGBY

NODE.JS AND EXPRESS - SERVING STATIC CONTENT

express static middleware

Node.js and Express - Serving Static Content

One of the great things about Node.js is that it has a built in HTTP server. This means you don't need Apache or nginx. This means serving a static site can be done in few lines of code. This article goes into how this can be achieved.

Express Static Middleware

Express has become the defacto Node.js web framework and it has great built in capabilities to serve static content. The nice thing is that not only can you serve static content you can also gzip compress it and cache it. But let's just start with the required package.json and a basic static server.

package.json

{

"name" : "ServeStaticContent",

"version" : "0.0.1",

"dependencies" : {

"express" : "3.x"

}

}

app.js - basic static server

var express = require('express');

var app = express();

app.use(express.static(\_\_dirname + '/public'));

app.listen(process.env.PORT || 3000);

The above code is very simple, it creates an Express server, adds the static middleware and finally starts listening on port 3000 or provided port. Modulus will provide the port using PORT environment variable if the application is hosted on the platform.

The static middleware handles serving up the content from a directory. In this case the 'public' directory is served up and any content (HTML, CSS, JavaScript) will be available. This means if the public directory looks like:

index.html

js - folder

main.js

css - folder

style.css

Then you can request the root route '/' and you'll get index.html file and if '/js/main.js' is requested you receive the the main.js file in the js folder. This is all expected from a static server. Now, you can do some more sophisticated configuration by passing a subpath into the app.use function.

app.use('/static', express.static(\_\_dirname + '/public'));

This allows for subpath content serving. This means if you went to '/static' you'd now get index.html.

Express Compress Middleware

Now most of the time you'll also want to make sure your static content is compressed using gzip. This can easily be done using the compress middleware that is bundled with Express. The main implementation detail is to make sure that the app.use call for compress is before any other middlewares (there are a few exceptions like logging).

The simplest form of this looks like the following:

var express = require('express');

var app = express();

// New call to compress content

app.use(express.compress());

app.use(express.static(\_\_dirname + '/public'));

app.listen(process.env.PORT || 3000);

That will return elements compressed with gzip if they're HTML, CSS, JavaScript, or JSON. You can double check this by using the Chrome debugger and then you can see the updated network response of the content to include gzip.

Express Static Middleware Cache-Control

The last thing that you may want to include is caching of the content. This again can be done very simply using the static middleware that you are already using. The code simply needs to be updated to include a max age directive in the Cache-Control header.

var express = require('express');

var app = express();

var oneDay = 86400000;

app.use(express.static(\_\_dirname + '/public', { maxAge: oneDay }));

app.listen(process.env.PORT || 3000);

Wrap-up

If you combine all the above pieces you'll end up with final static server code that looks like:

var express = require('express');

var app = express();

var oneDay = 86400000;

app.use(express.compress());

app.use(express.static(\_\_dirname + '/public', { maxAge: oneDay }));

app.listen(process.env.PORT || 3000);

That's all it takes to have a nice easy to use static server using Express and Node.js. You can grab this project on Github to take a look at all the code yourself. Feel free to leave a comment if you have any questions.

WHAT IS MODULUS?

Modulus is a home for web applications. Node.js hosting, integrated MongoDB, and performance metrics all in one place. It’s free to get started.

GET $15 FOR SIGNING UP

SHARE THIS ARTICLE

ABOUT

Modulus is a premier Node.js hosting platform that provides a complete technology stack for application developers. This includes custom SSL, WebSockets, MongoDB, statistics, and more.

GET STARTED FOR FREE

TAGS

Modulatorprismicbuildnodeconf&yetdbJavaScriptasset pipelinevideostartupguidestaticdeploymentjoyenthackathon

RELATED POSTS

An Absolute Beginner's Guide to Node.js

Node.js and Express - Basic Authentication

Node.js and Express - Sessions

Node.js and Express - Creating a Rest API

Node.js and MongoDB - Getting Started with Mongoose

NEWSLETTER SIGNUP

The Modulus monthly newsletter brings you exclusive content only available to subscribers and the best Node.js material from around the web.

CONTACT US

@OnModulus

facebook.com/OnModulus

C:\Users\User1\.WebStorm8\system\extLibs\nodejs-v0.10.32-src\core-modules-sources\lib\events.js

MongoDB

===========

Run MongoDB:

c:\mongodb\bin\mongod.exe

Connect using the mongo client:

c:\mongodb\bin\mongo