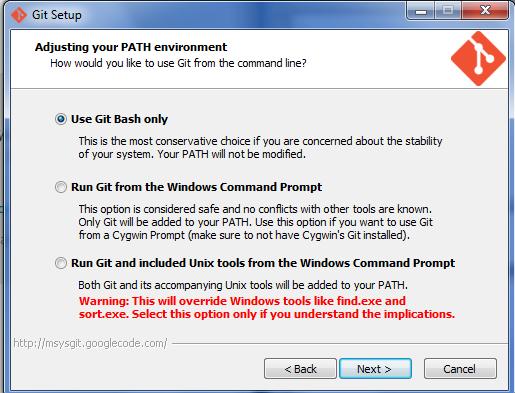
**KIT DE DESENVOLVIMENTO JAVASCRIPT**

**GIT**

* Instalando Git
  + Linux:
    - sudo apt-get install git
    - Teste com o commando git no terminal
  + Windows: Fazer download neste link <http://git-scm.com/downloads>
    - Executar o instalador
    - Siga instalando até esta tela e mantenha *Use Git Bash only* instalado



* + - Termine de instalar.
* Configuração
  + Abrir o Git Bash no windows ou Terminal no Linux.
    - git config --global user.name <<seu nome>
    - git config --global user.email [<<seu](mailto:johndoe@example.com) email>>
    - Para cores: git config --global color.ui auto
  + Para gerar chaves de usuário e configurar o github para aceitá-la siga o link <https://help.github.com/articles/generating-ssh-keys> ou consulte o anexo.
  + Selecione o seu sistema operacional e siga os passos do tutorial
  + Para finalizar faça uma primeira conexão no git com:
    - ssh [git@github.com](mailto:git@github.com)
    - Respondam YES caso pergunte se deseja lembrar a chave de máquina
  + Após esse procedimento você poderá clonar projetos através do SSH do Github, e dar pull e push no Git sem colocar sua senha.

FERRAMENTA DE BUILD

No Linux:

* Instale o Node Js com o comando
  + sudo apt-get install node
* Após a instação anterior , instale o NPM, com o comando:
  + Sudo apt-get install npm
* Instale o browser headless PhantomJS
  + Sudo npm install phantomjs -g
* Instale via NPM a ferramenta de build Grunt
  + Sudo npm install grunt –g

No Windows:

* Entre no site nodejs.org e clique em Install. Isso irá baixar o instalador de Node.js para Windows
* Abra o arquivo do instalador e faça instalação
  + - Configurar uma variável PATH com o endereço do NPM nas variáveis de ambiente do usuário local.
    - O conteúdo da variável deve ser o endereço para o NPM conforme o exemplo: **C:\Users\joao\AppData\Roaming\npm\**
* Abra o Prompt de Comando, e instale o grunt e o browser headless PhantomJS
  + npm install grunt –g
  + npm install phantomjs –g
* Na instalação do Phantom, ao final será mostrado o diretório em que se encontra o arquivo phantomjs.exe, copie este diretório.
  + Clique Iniciar >> botão direito em Computador >> Propriedades
  + No canto esquerdo selecione Configurações Avançadas do Sistema
  + Na aba Advanced, clique no botão Variáveis de Ambiente
  + Na seção Variáveis de Sistema, localize a variável Path, duplo clique e cole no final de Valor da Variável o seguinte caminho
    - <<caminho do phantom>>\;
    - Exemplo: **C:\Users\joao\AppData\Roaming\npm\node\_modules\phantomjs\lib\phantom\;**
    - Não esqueça do ponto-e-virgula no final e não dê espaço após o ponto-e-vírgula anterior.
  + Teste se o phantomjs.exe funciona, reiniciando o prompt de comando e usando e invocando
    - phantomjs
  + Teste o funcionamento do grunt, criando um projeto básico (abaixo) e rodando os testes (grunt qunit)

Projeto básico

* No prompt de comando, navegue até uma pasta para o projeto javascript , e inicialize o seu projeto padrão de build com jquery:
  + Grunt.cmd init:jquery
* Essa é a estrutura mínima para rodar testes e gerenciar dependências.
* As dependências devem ser inseridas no arquivo package.json na raiz do projeto
* Pode-se customizar o grunt para usar diferentes bibliotecas de testes, como jasmine, mocha, etc
  + Para isso altere o arquivo grunt.js na base do projeto.
  + A biblioteca padrão de testes no grunt é o qUnit (grunt.cmd qunit)

**JENKINS**

* Instalação
  + Linux:
    - sudo apt-get install jenkins
    - Teste entrando na URL http://localhost:8080 , deverá entrar na página principal do jenkins
  + Windows:
    - Baixe a versão nativa para windows no site <http://jenkins-ci.org/>
    - Descompacte para um diretório desejado
    - Rode o arquivo setup.exe
    - Finalize a instalação
    - Teste entrando na URL http://localhost:8080 , deverá entrar na página principal do jenkins
* Integração do Jenkins com Git
  + Linux:
    - Entre no terminal
    - Mude para usuário jenkins
      * Sudo su - -s /bin/bash jenkins
    - Configure:
      * git config –global user.email “<<email do usuario>>”
      * git config –global user.name “jenkins”
      * Crie uma nova chave para o usuário jenkins conforme anexo
      * Não esqueça de adicionar a chave no github
  + Windows:
    - Configure as variáveis de ambiente seguindo os passos abaixo:

## Setup your environment variables

General hint: Avoid spaces in environment paths

Mainly, you will need:

* GIT\_HOME => Folder where your git.exe is located (e.g. C:\Program Files (x86)\Git\bin )
* HOME => Folder where your SSH Keys are located (e.g C:\Users\OnCast Marketing)
* PATH => Add the folder where your plink.exe is located

Once this is done, make sure you restart your consoles and the jenkins service.

* Configuração Jenkins
  + Vá em Jenkins, no canto superior esquerdo >> Manage Jenkins >> Manage Plugins
  + Vá na aba *Available* >> Selecione *o GitHub Plugin* >> Clique *Install without restart*
* Configurando novo projeto Jenkins
  + Na página inicial do Jenkins, selecione no menu a esquerda *New Job*
  + Dê um nome ao projeto no campo *Job Name*
  + Escolha Build a free-style software project e clique em ok OK
  + Na próxima página em Source Code Management selecione Git
  + Cole no campo Repository URL o caminho do repositório da aplicação (ex: [git@github.com:oncast-labs/javascript-bootstrap.git](mailto:git@github.com:oncast-labs/javascript-bootstrap.git))
  + Configurar build:
    - No Linux: selecione “Run Shell” e digite: npm install; grunt
    - No Windows: Selecione “Run CMD” e digite: npm install, de um enter e digite grunt.cmd
  + Clique em Save no final da página
  + Teste rodando o build de projeto. O jenkins deverá baixar o projeto e construí-lo

**Kit para Desenvolvimento**

**Git**

* Seguir o mesmo procedimento do kit da máquina de Integração Contínua

**NPM e Grunt**

* Seguir o mesmo procedimento do kit da máquina de Integração Contínua

**Sublime Text**

* Baixar o editor no link: <http://www.sublimetext.com/>
* Extraír o zip e executar.

Tutorial Github: Generating SSH Keys

* [**MAC**](https://help.github.com/articles/generating-ssh-keys#platform-mac)
* [**WINDOWS**](https://help.github.com/articles/generating-ssh-keys#platform-windows)
* [**LINUX**](https://help.github.com/articles/generating-ssh-keys#platform-linux)
* [**ALL**](https://help.github.com/articles/generating-ssh-keys#platform-all)

If you have decided not to use the [recommended HTTPS method](https://help.github.com/articles/set-up-git), we can use SSH keys to establish a secure connection between your computer and GitHub. The steps below will walk you through generating an SSH key and then adding the public key to your GitHub account.

**Step 1: Check for SSH keys**

*Have an existing keypair you'd like to use? You can skip to****Step 4****.*

First, we need to check for existing ssh keys on your computer. Open up Terminal and run:

cd ~/.ssh# Checks to see if there is a directory named ".ssh" in your user directory

If it says "No such file or directory" skip to **step 3**. Otherwise continue to **step 2**.

**Step 2: Backup and remove existing SSH keys**

Since there is already an SSH directory you'll want to back the old one up and remove it:

ls# Lists all the subdirectories in the current directory

# config id\_rsa id\_rsa.pub known\_hosts

mkdir key\_backup# Makes a subdirectory called "key\_backup" in the current directory

cp id\_rsa\* key\_backup# Copies the id\_rsa keypair into key\_backup

rm id\_rsa\*# Deletes the id\_rsa keypair

**Step 3: Generate a new SSH key**

To generate a new SSH key, enter the code below. We want the default settings so when asked to enter a file in which to save the key, just press enter.

ssh-keygen -t rsa -C "*your\_email@youremail.com*"# Creates a new ssh key using the provided email

# Generating public/private rsa key pair.

# Enter file in which to save the key (/home/*you*/.ssh/id\_rsa):

Now you need to enter a passphrase.

[**Why do passphrases matter?**](https://help.github.com/articles/generating-ssh-keys)

# 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 /home/*you*/.ssh/id\_rsa.

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

# The key fingerprint is:

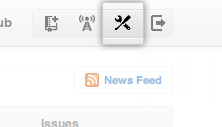
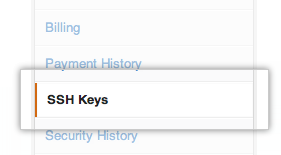
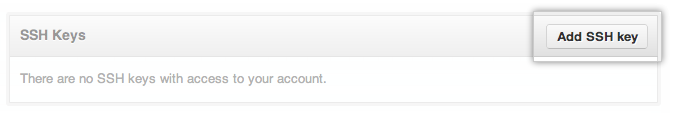
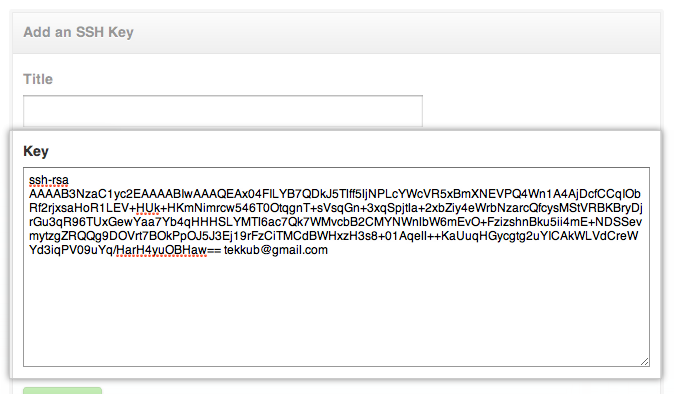
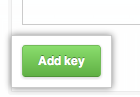
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db *your\_email@youremail.com*

**Step 4: Add your SSH key to GitHub (WINDOWS)**

Run the following code to copy the key to your clipboard.

clip < ~/.ssh/id\_rsa.pub# Copies the contents of the id\_rsa.pub file to your clipboard

**Be warned:** it is important to copy the key exactly without adding newlines or whitespace. Thankfully the clip command makes it easy to perform this setup perfectly.

1. Go to your [Account Settings](https://github.com/settings)
2. Click ["SSH Keys"](https://github.com/settings/ssh) in the left sidebar
3. Click "Add SSH key"
4. Paste your key into the "Key" field
5. Click "Add key"
6. Confirm the action by entering your GitHub password

**Step 4: Add your SSH key to GitHub (LINUX)**

Run the following code to copy the key to your clipboard.

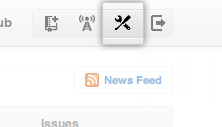
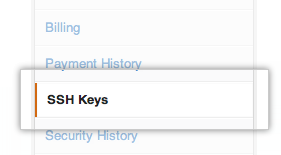
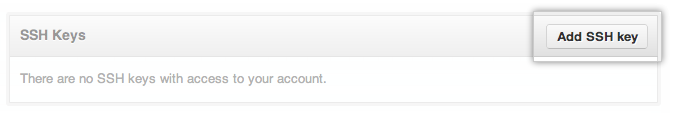
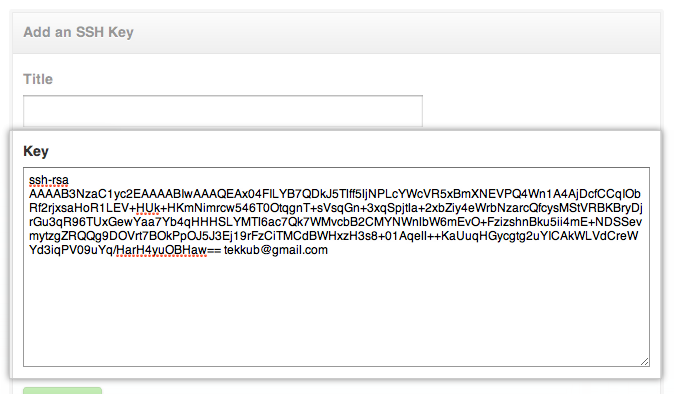
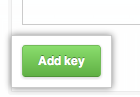
sudo apt-get install xclip

# Downloads and installs xclip

xclip -sel clip < ~/.ssh/id\_rsa.pub

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

**Be warned:** it is important to copy the key exactly without adding newlines or whitespace. Thankfully the xclip command makes it easy to perform this setup perfectly.

1. Go to your [Account Settings](https://github.com/settings)
2. Click ["SSH Keys"](https://github.com/settings/ssh) in the left sidebar
3. Click "Add SSH key"
4. Paste your key into the "Key" field
5. Click "Add key"
6. Confirm the action by entering your GitHub password

**Step 5: Test everything out**

To make sure everything is working you'll now SSH to GitHub. When you do this, you will be asked to authenticate this action using your password, which for this purpose is the passphrase you created earlier. Don't change the git@github.com part. That's supposed to be there.

ssh -T [git@github.com](mailto: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 matches the one here and type "yes".

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

# provide shell access.

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

If you see "access denied" please consider using [HTTPS](https://help.github.com/articles/set-up-git) instead of SSH. If you need SSH start at[these instructions](https://help.github.com/articles/error-permission-denied-publickey) for diagnosing the issue.