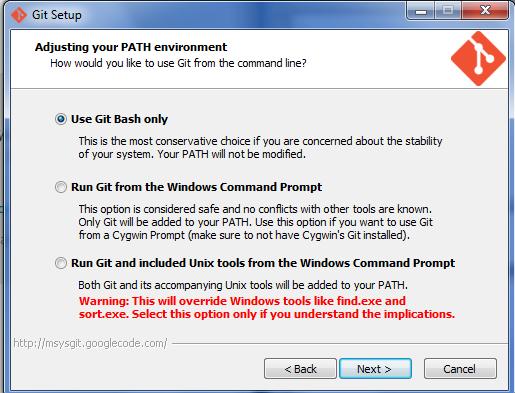
**KIT DE DESENVOLVIMENTO JAVA**

**Tempo de preparação Windows: 90 minutos**

**Tempo de Preparação Linux: 60 minutos**

**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.

**MAVEN**

* Instalação:
  + No Linux:
    - sudo apt-get install maven2
    - Teste com o commando *mvn* no terminal
  + No Windows: baixar o binário mais recente zipado no site <http://maven.apache.org/download.html>
    - Descompactar para diretório escolhido no sistema
    - Vamos configurar agora a variável de ambiente:
      * 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, crie uma nova variável, coloque como Nome da Variável “JAVA\_HOME” (sem as aspas) , e Valor da Variável o seguinte caminho
        + <<caminho do java no PC>>
        + Exemplo: C:\Program Files (x86)\Java\jdk1.7.0\_09
        + Teste o funcionamento do java executando o comando *javac* no Prompt de Comando
      * Na seção Variáveis de Sistema, localize a variável Path, duplo clique e cole no final de Valor da Variável o seguinte caminho
        + <<caminho do maven>>\bin\;
        + Exemplo: c:\ apache-maven-3.0.4\bin
    - Teste o funcionamento do maven abrindo o Prompt de Comando, e invocando o commando ***mvn***

**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

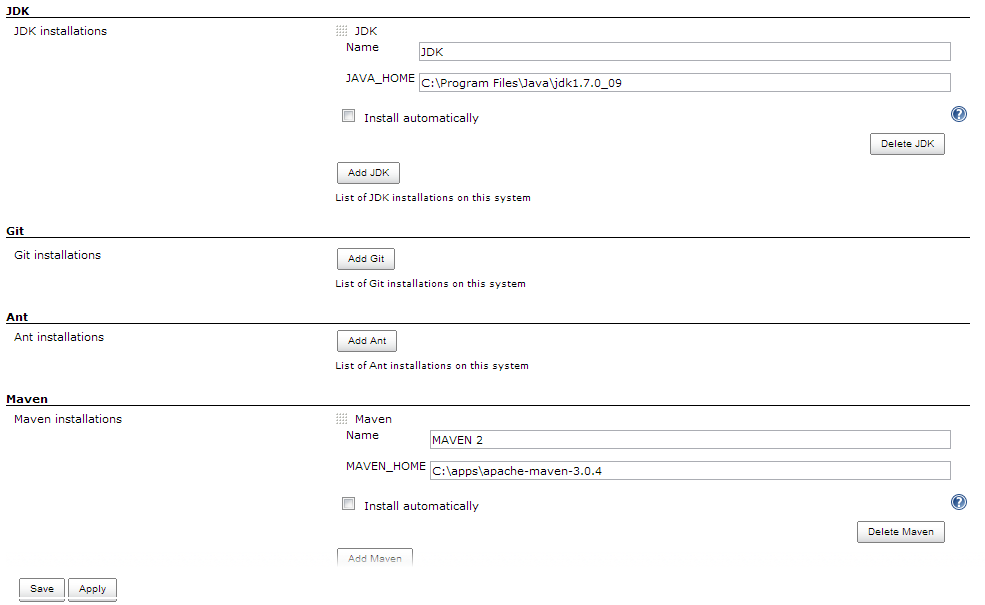
General hint: Avoid spaces in environment paths

Mainly, you will need:

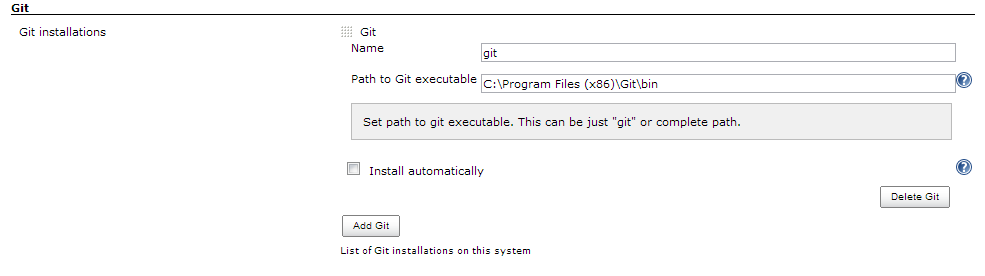
* GIT\_HOME => Folder where your git.exe is located (e.g. C:\Program Files (x86)\Git\bin )
* HOME => Folder where your SSH Keys are located (e.g C:\Users\OnCast Marketing\.ssh)
* PATH => Add the folder where your plink.exe is located

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*
  + Vá em *Manage Jenkins* >> Configure Systems >> em JDK clique *JDK Installations*
  + Desmarque *Install automatically,* adicione um nome para a JDK, adicione o caminho da JDK instalada (ex: C:\Program Files\Java\jdk1.7.0\_09 ou /usr/lib/java-7-openjdk) .
  + Ainda em Manage Jenkins >> Configure Systems >> em Maven clique Add Maven
  + Desmarque *Install automatically,* adicione um nome para o Maven, adicione o caminho do Maven instalado (ex: C:\apache-maven-3.0.4 ou /usr/share/maven2)
  + Veja imagem abaixo:



* + Faça o mesmo para configurar o executável Git no Windows
  + Desmarque *Install automatically,* adicione um nome para o Git, adicione o caminho do Git instalado (ex: C:\Program Files (x86)\Git)



* + No final da pagina clique em *Save*
* 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 maven2/3 project, clique 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/zion-security.git)
  + Clique em *Save* no final da página
  + Teste rodando o build de projeto. O jenkins deverá baixar o projeto e construí-lo

**SONAR**

* Instalação
  + Linux:
    - Vá no link <http://www.sonarsource.org/downloads/> e baixe o arquivo compactado do Sonar
    - Descompacte em diretório desejado
    - Siga as instruções para criação de serviço Linux em anexo
    - Vá para <http://localhost:9000> para verificar se está rodando
  + Windows:
    - Vá no link <http://www.sonarsource.org/downloads/> e baixe o arquivo compactado do Sonar
    - Descompacte em diretório desejado
    - Voce pode colocar o Sonar para rodar como serviço windows:
      * Vá no subdiretório bin/windows-x86-32 ou bin\windows-x86-64
      * Clique com o outro botão em /InstallNTService.bat
      * Clique em *Executar como administrador/ Run as administrator*
    - Ou pode rodar diretamente
      * Vá no subdiretório bin/windows-x86-32 ou bin\windows-x86-64
      * Clique duas vezes em *StartSonar.bat*
    - Vá para <http://localhost:9000> para verificar se está rodando
* Intregrando Jenkins e Sonar
  + Vá em Jenkins, no canto superior esquerdo >> Manage Jenkins >> *Manage Plugins*
  + Vá na aba *Available* >> Selecione *o Sonar Plugin* >> Clique *Install without restart*
  + Só para Linux: vá em *Manage Jenkins >> Manage Plugins >>* Seleciona *Maven 2 Project Plugin >>* Clique *Install without restart*
  + Vá em *Manage Jenkins* >> *Configure Systems* >> em Sonar clique Add Sonar
  + Dê um nome para seu Sonar, e configure o campo Server URL como http://localhost:9000
  + No final da pagina clique em *Save*
* Utilizando Sonar no projeto
  + Para utilizar o Sonar no seu projeto, você deve ir no nome do Job >> *Configure* >> *Post-build Actions* >> selecione *Sonar* >> clique em *Save* no fim da página

**GIT PARA DESENVOLVIMENTO**

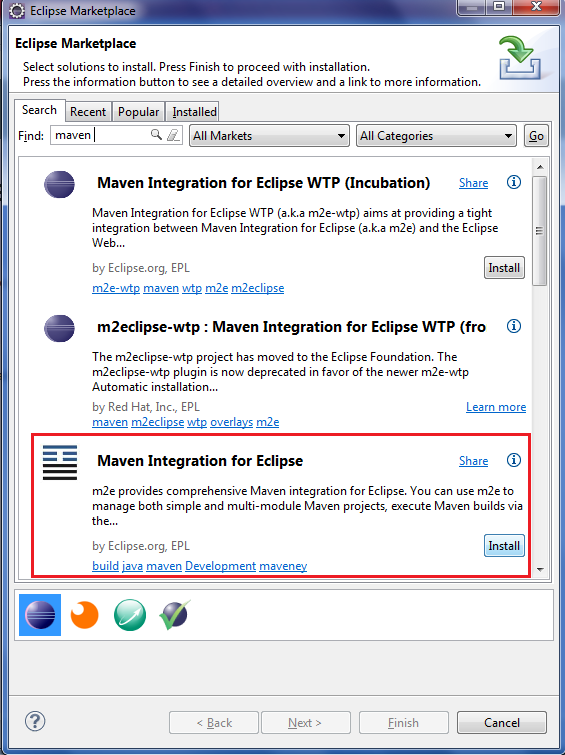
* Iniciando projeto novo sem conteúdo inicial
  + Abrir o Git Bash no windows ou Terminal no Linux.
  + Crie uma pasta para seu projeto com o comando:
    - Mkdir <nome pasta>
  + Navegue via terminal ou bash para a pasta e use o comando git para iniciar o repositório:
    - Git init
* Iniciando projeto novo com conteúdo inicial
  + Abrir o Git Bash no windows ou Terminal no Linux.
  + Navegue via terminal ou bash para a pasta do projeto e use o comando git para iniciar o repositório:
    - Git init
  + Adicione todos os arquivos e diretórios do diretório ao commit
    - Git add .
  + Dê commit
    - Git commit –m “*mensagem de commit explicando o que foi feito*”
  + Configure o servidor github que vai o projeto, substituindo *usuario* pelo nome do usuário e *repo* pelo nome do repositório.
* git remote add origin <https://github>.com/*usuario*/*repo*.git
  + Envie os commits ao servidor
    - Git push
* Clonando projeto existente no servidor
  + Para clonar projeto já existente, no terminal ou bash, navegue até a pasta em que deseja clonar o projeto e use o comando:
    - git clone < endereço git do projeto, visível no github >
    - Ex: git clone [git@github.com:oncast-labs/oncast-cms.git](mailto:git@github.com:oncast-labs/oncast-cms.git)

**MAVEN PARA DESENVOLVIMENTO**

* Com os passos de instalação realizados anteriormente, você já tem seu sistema pronto para usar o Maven.
* Alguns comandos mais utilizados (sendo mvn <<comando>>)
* compile – compila código fonte
* test – testa o código compilado usando framework de testes.
* package – empacotamento em arquivo distribuível, como .jar e .war
* integration-test – roda testes de integração
* install – cria o pacote em repositório local, para uso em outras dependências
* Se você rodar qualquer dos comandos , todos os comandos anteriores serão executados naturalmente.

**ECLIPSE**

* Instalação
  + Linux
    - Entre em <http://www.eclipse.org/downloads/>, faça download de *Eclipse IDE for Java EE developers* para a versão do seu Linux, 32 bits ou 64 bits.
    - Descompacte em diretório de sua preferência
    - Entre no diretório do eclipse e acesse o eclipse
    - Escolher o diretório para workspace e selecionar ok
  + Windows
    - Entre em <http://www.eclipse.org/downloads/>, faça download de *Eclipse IDE for Java EE developers* para a versão do seu Windows, 32 bits ou 64 bits.
    - Descompacte em diretório de sua preferência
    - Entre no diretório do eclipse e acesse o eclipse.exe
    - Escolher o diretório para workspace e selecionar ok
* Configuração do plugin do maven
  + Acessar o menu Help >> Eclipse Marketplace e procure por maven conforme a figura abaixo:



* Selecionar Install, aceitar as condições de instalação e quando for perguntado se deseja reiniciar o Eclipse, selecione que sim

PROJETO BOOTSTRAP

* No github há disponível um projeto Maven com as dependências necessárias para um projeto padrão Java Web:
* Em Maven, através do pom.xml
* Com Junit (testes), Flyway (migrations), DBUnit (integration tests), Mockito (mocks), Cobertura (cobertura de testes)
* Endereço Github: git@github.com:oncast-labs/bootstrap-java.git

# [Running Sonar as a Service on Linux](http://docs.codehaus.org/display/SONAR/Running+Sonar+as+a+Service+on+Linux)

**Create the file /etc/init.d/sonar with this content:**

|  |
| --- |
| #!/bin/sh  #  # rc file for Sonar  #  # chkconfig: 345 96 10  # description: Sonar system (www.sonarsource.org)  #  ### BEGIN INIT INFO  # Provides: sonar  # Required-Start: $network  # Required-Stop: $network  # Default-Start: 3 4 5  # Default-Stop: 0 1 2 6  # Short-Description: Sonar system (www.sonarsource.org)  # Description: Sonar system (www.sonarsource.org)  ### END INIT INFO    /usr/bin/sonar $\* |

**Register Sonar at boot time (Ubuntu, 32 bit):**

|  |
| --- |
| sudo ln -s $SONAR\_HOME/bin/linux-x86-32/sonar.sh /usr/bin/sonar  sudo chmod 755 /etc/init.d/sonar  sudo update-rc.d sonar defaults |

**Register Sonar at boot time (RedHat, CentOS, 64 bit):**

|  |
| --- |
| sudo ln -s $SONAR\_HOME/bin/linux-x86-64/sonar.sh /usr/bin/sonar  sudo chmod 755 /etc/init.d/sonar  sudo chkconfig --add sonar |

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:

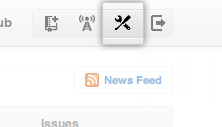
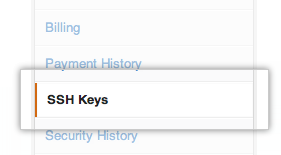
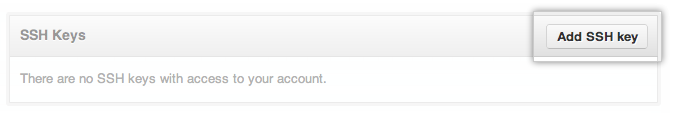
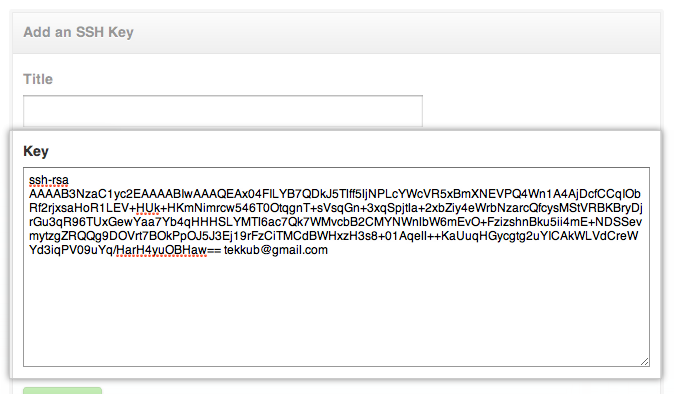
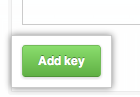
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db *your\_email@youremail.com*

**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.

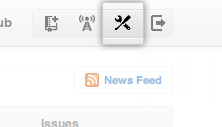
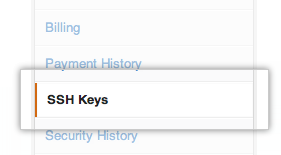
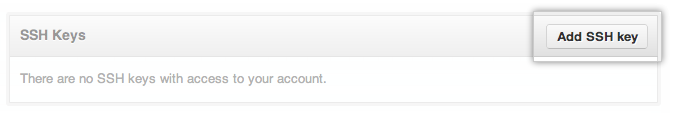
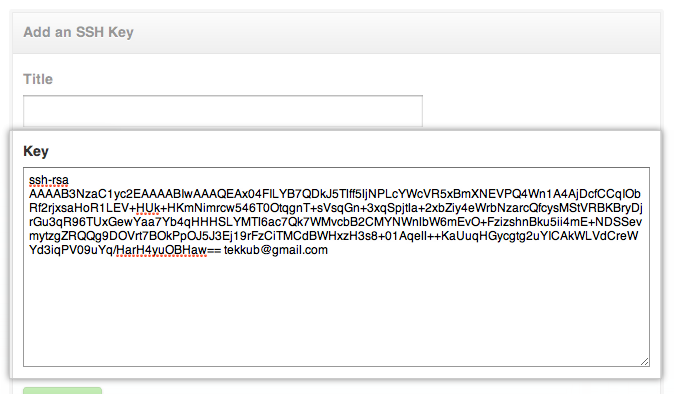
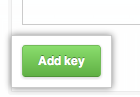
sudo apt-get install xclip

# 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.