If you are using this presentation from the OpenOffice file instead of the PDF:

For best formatting, please install the FreeSansBold.ttf font under openoffice/share/font or windows/fonts:

http://thewoolleyweb.com/ci_for_the_web_2.0_guy_or_gal/tools/font/FreeSansBold.ttf

(even then, some OpenOffice platforms may require you to do outline -> select all -> and pick the font. But hey, it's free...)

Welcome! If you want to follow along, borrow a flash drive, copy the contents to your drive, and see the README. Or, download from:

thewoolleyweb.com/ci_for_the_web_2.0_guy_or_gal

CI for the Web 2.0

/G(uy|al)/

Obligatory Boiler Plate

Who

Chad Woolley
thewoolleyman @ gmail.com

Pivotal
Labs
.com

Who are YOU? CI? Linux? Virtualization? Java? Ruby? JsUnit? Selenium?

What

CI == Continuous Integration

Martin Fowler - Seminal CI Article

Running all your tests on every commit

Automatically

How

Takahashi Method == Big Font!

Focused on how to install and make everything work together, not on details of how to use the tools

Agenda:

1. Code: The simplest tutorial that could POSSIBLY work

Coding Tasks Outline

A. Install Linux on VMWare

B. Install Prereqs: ruby, java, mysql, svn, ant, alternate browser

C. Create sample Ruby on Rails Project

D. cruisecontrol.rb setup

E. JsUnit Setup

F. Selenium Setup

2. Gettin' Fancier

3. Gotchas

4. Questions

Tools Used

Cross-Platform, Mostly* Free

* VMware is not free on all platforms

VMware

Parallels is a Virtualization Alternative

Or, you can skip Virtualization and install Ubuntu directly on a spare PC. Just burn the ISO image to a CD.

Ubuntu Linux

cruisecontrol.rb

JsUnit

Selenium

There is a lot of material in this presentation

We will move FAST

Maybe too fast for you to follow along during the preso (sorry!)

But it's all on the slides

You can yell “Bingo” if you finish it before I do.

Intended to be comprehensive, easily repeatable, generic, cross-platform

Contains everything* you need to try this on a real project

* “everything” except the stuff that doesn't work on your project or environment. Error messages and Google are your friend :)

As a matter of fact, it almost certainly won't work perfectly for you. Integrating this stuff is hard, and new problems arise as tools and libraries evolve. Embrace the ~~bleeding~~ cutting edge, keep a positive attitude, and help fix bugs.

It's OK to sit back and watch

Try it at your home or workplace, at your own pace

Live!

No Hand Waving

(Warning: Obligatory lame attempt at humor coming up)

Their WILL be typos!

You down with OCD?

Then you'll know me!

Just please don't be “That Guy” (or Gal)!

You know, “That Guy” who stands up and wants to expound on irrelevant minutiae during the middle of a presentation...

Nitpicks and Linux hints Welcome...

...over beer, AFTER the presentation

...but seriously, if you are a bit OCDish, you might make a good CI G(uy|al) - because there's a lot of moving parts that all have to integrate...

...Continuously!

1. Time to Code!

A. Install Linux on VMWare

No time to install Linux live, but VMWare and images are on USB Keys

My Barebones Linux VM Setup:

Base:
VMWare on Macbook Pro 17”
Ubuntu 7.04 desktop VM from ISO
VMware Tools installed

Optional:
Change resolution (1680x1050)
Mouse Acceleration and Sensitivity
Terminal scrollback

Everything should work on pretty much any modern Unix distro

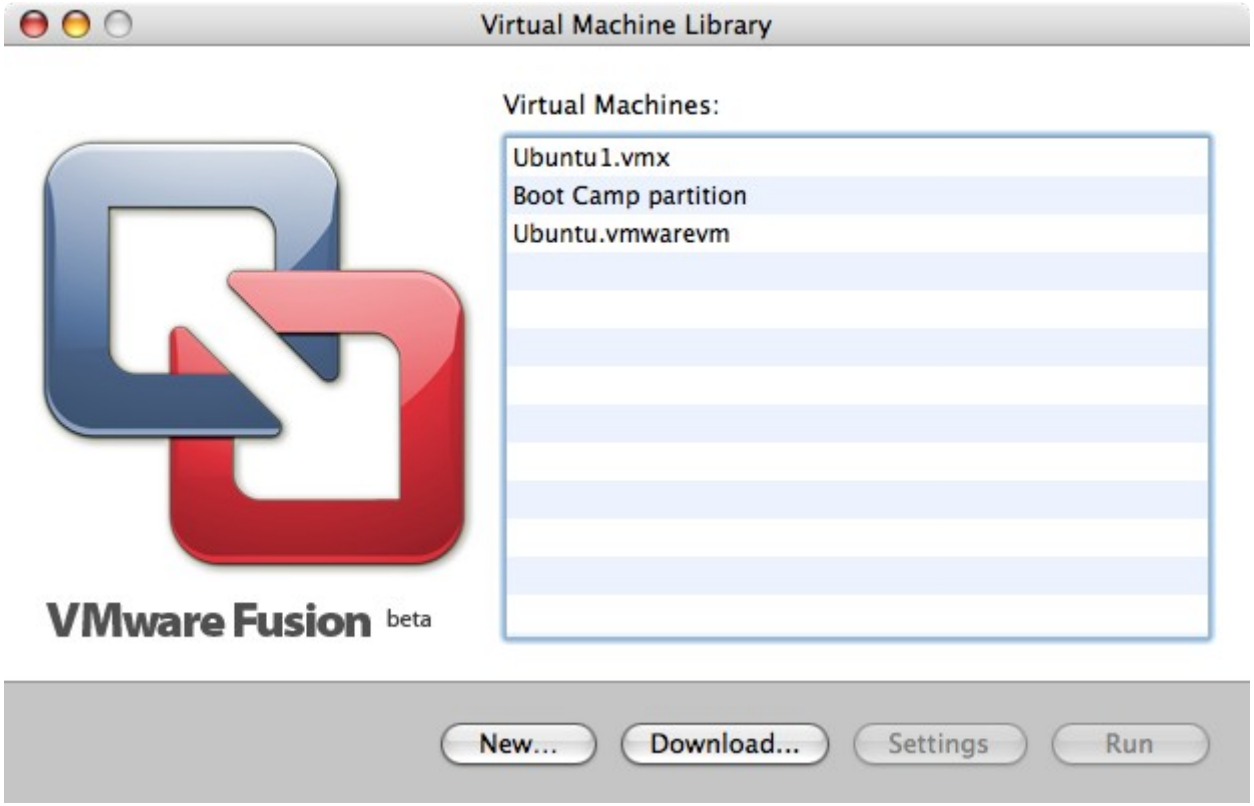
Following are screenshots and instructions to set up basic Ubuntu on VMware

We will skip them for now, but you can use them as a guide when you try it later

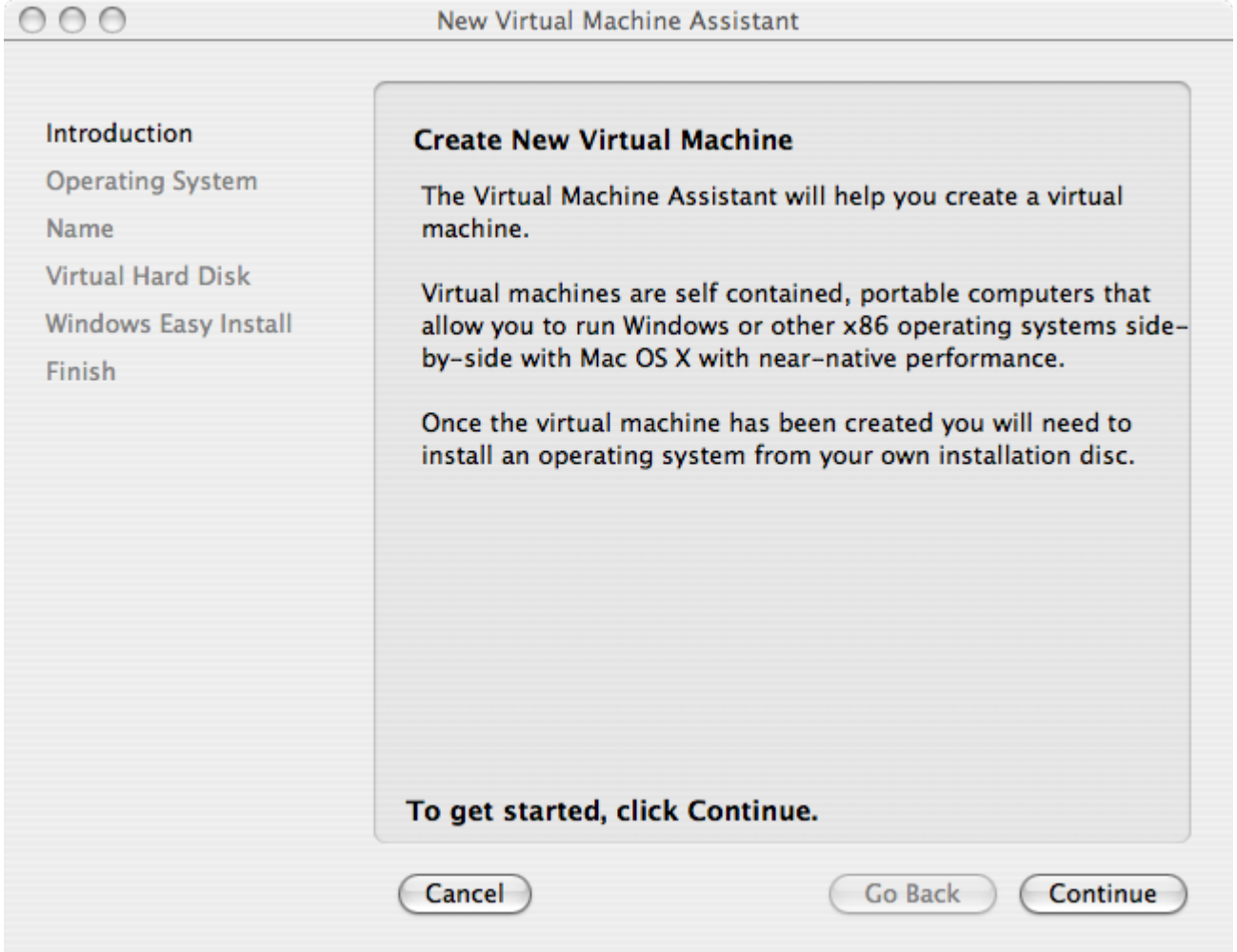
Original screenshots in /presentation /screenshots if these are too small to read

VMware Mac Setup: /presentation /screenshots /01a_mac_vmware_fusion_screenshots

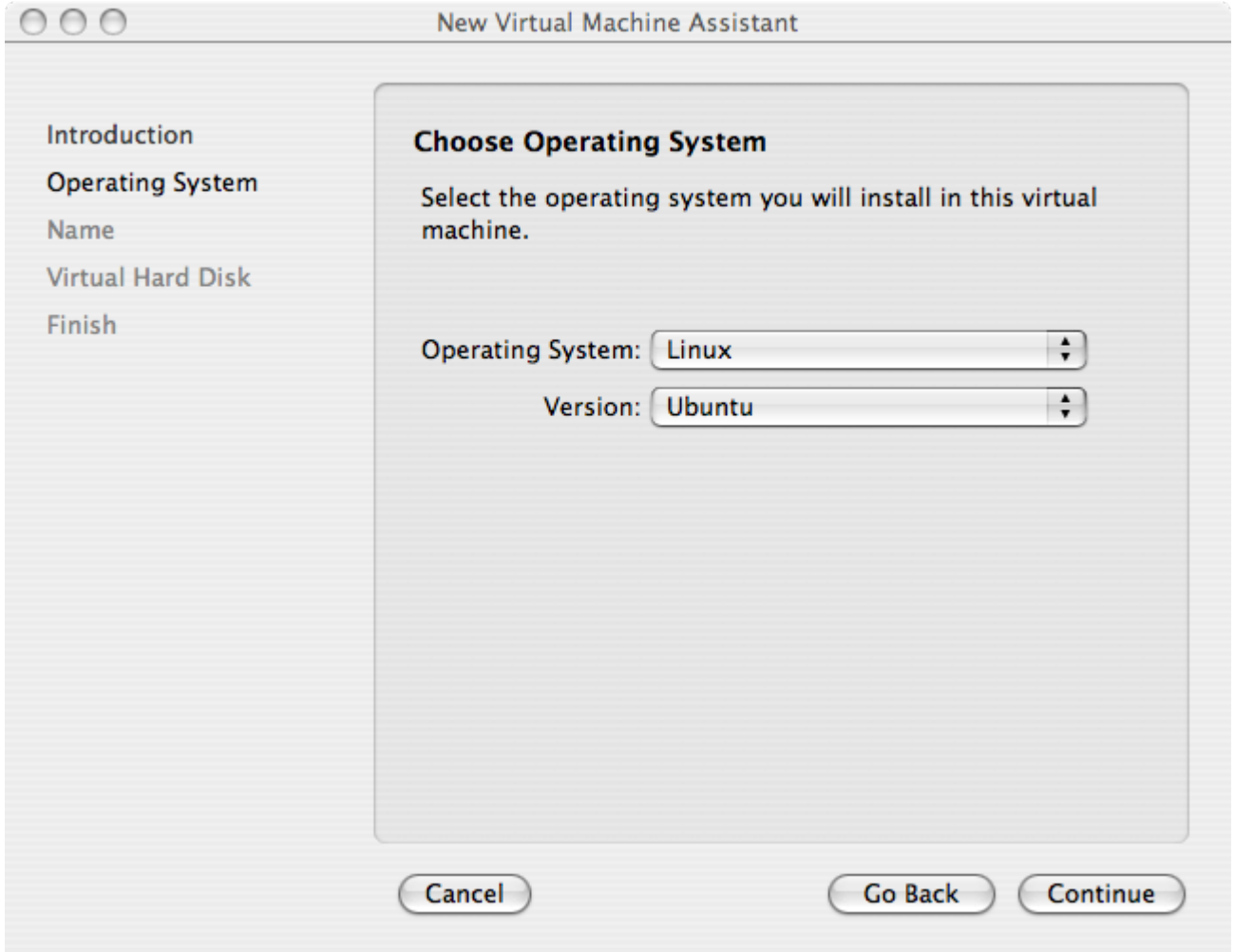
01_Virtual_Machine_Library.png



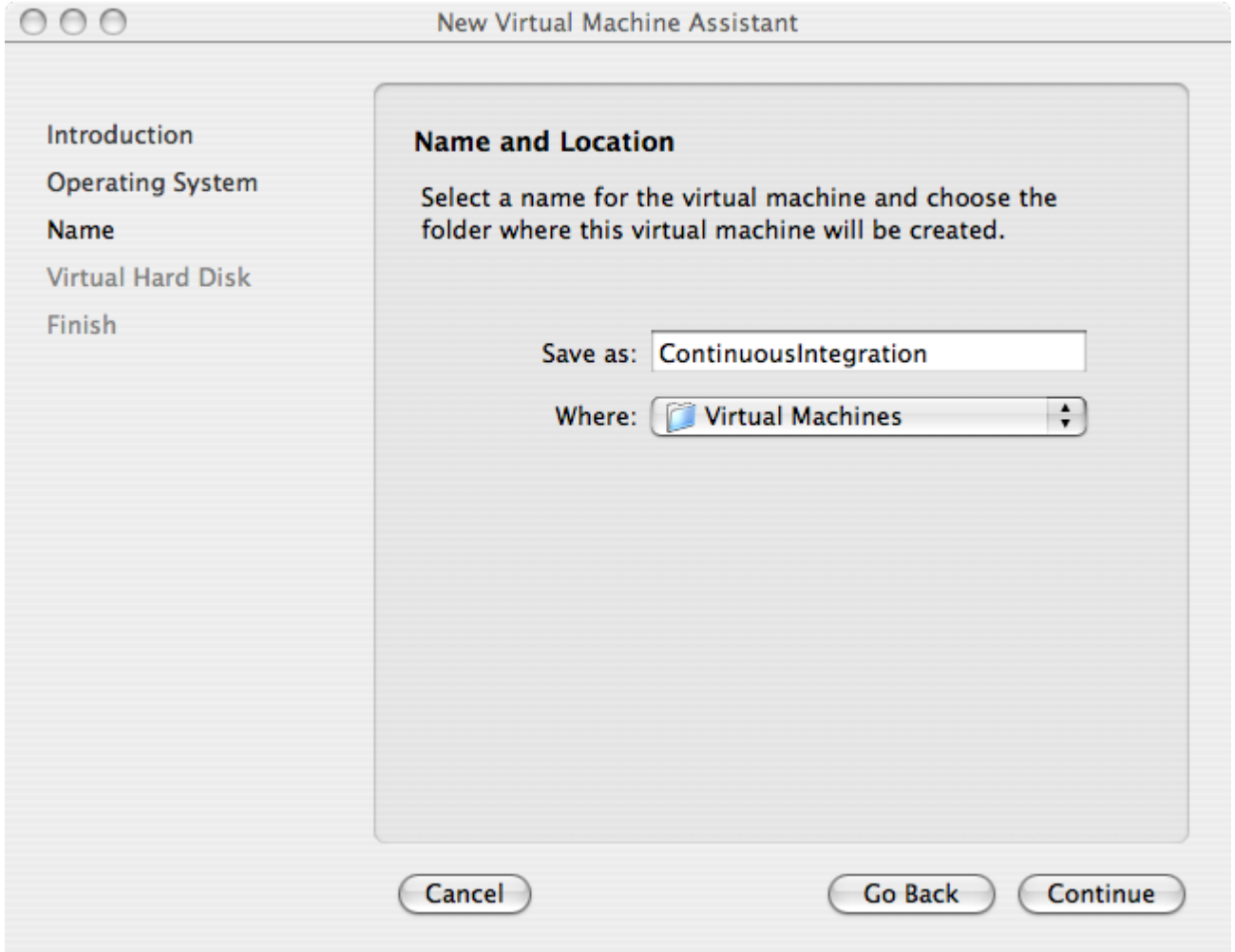
02_Create_New_Virtual_Machine.png



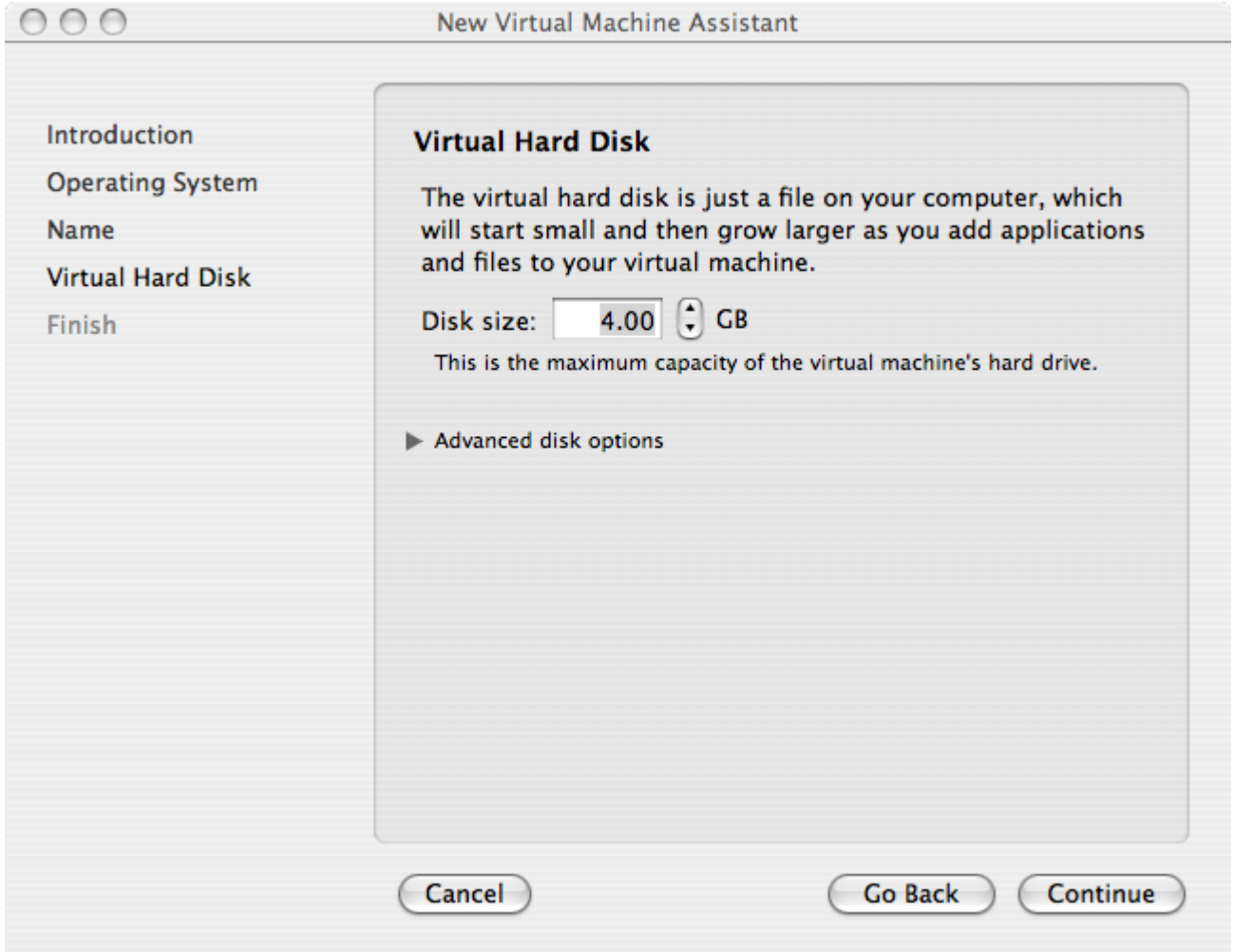
03_Choose_Operating_System.png



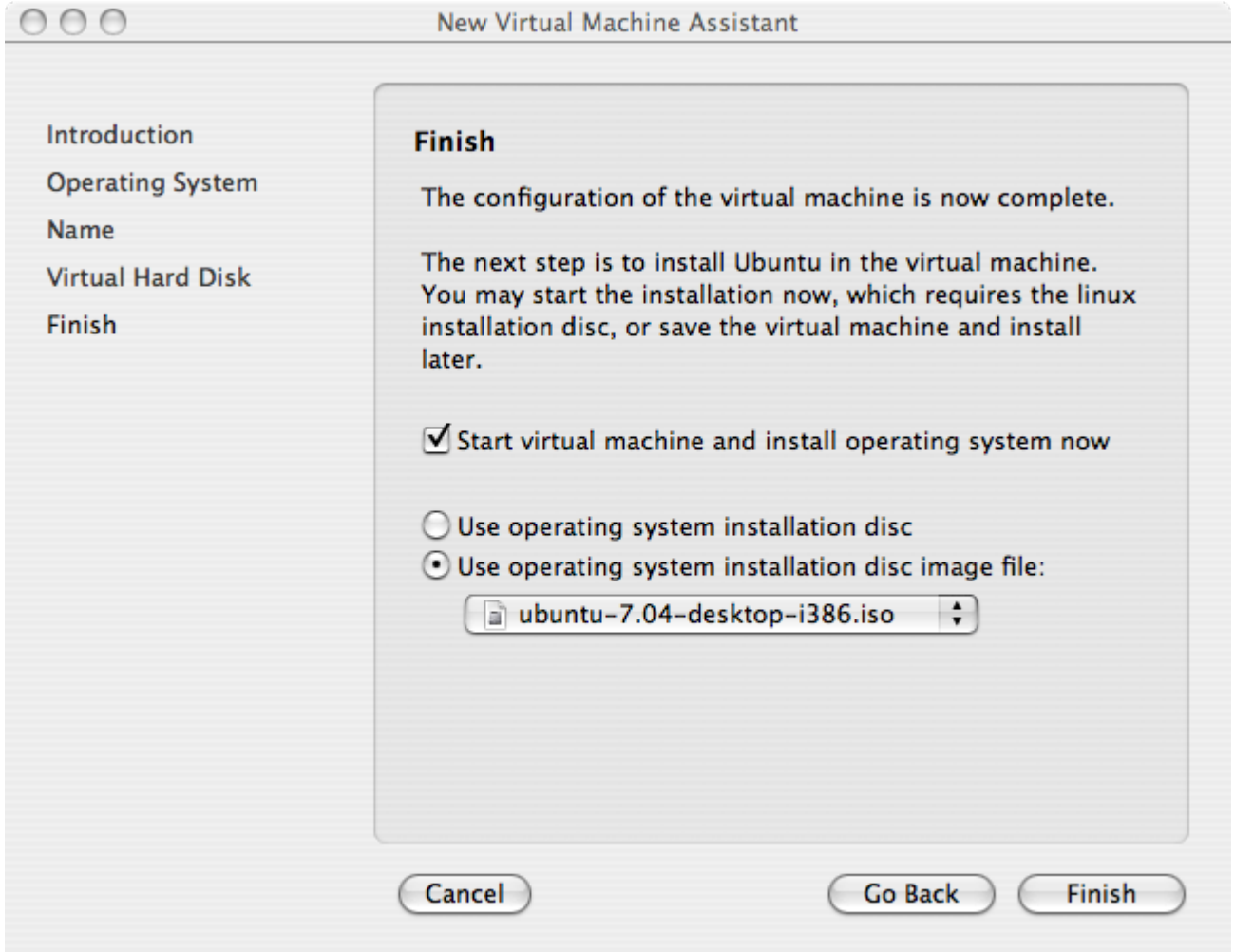
04_Name_and_Location.png



05_Virtual_Hard_Disk.png

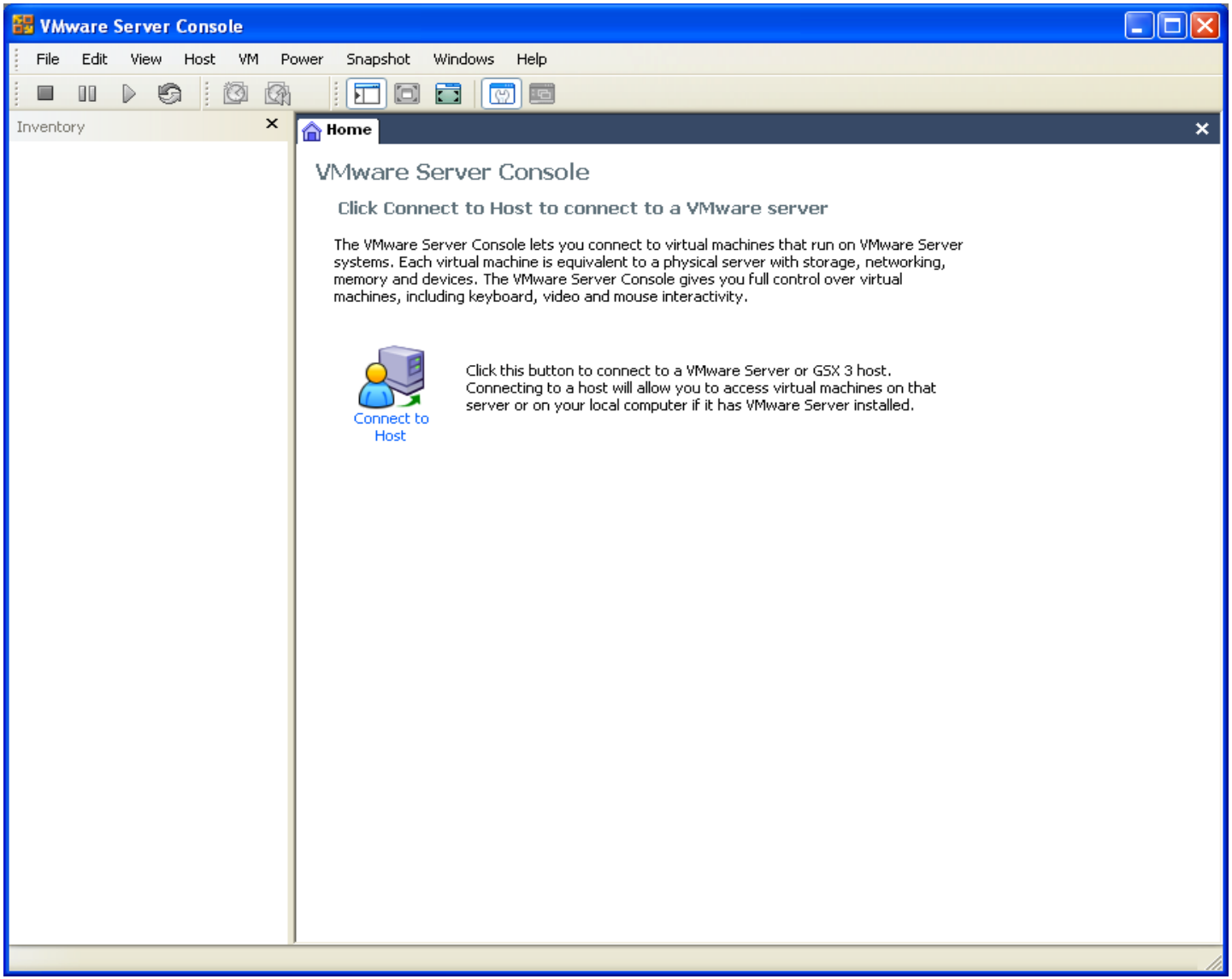


06_Finish.png

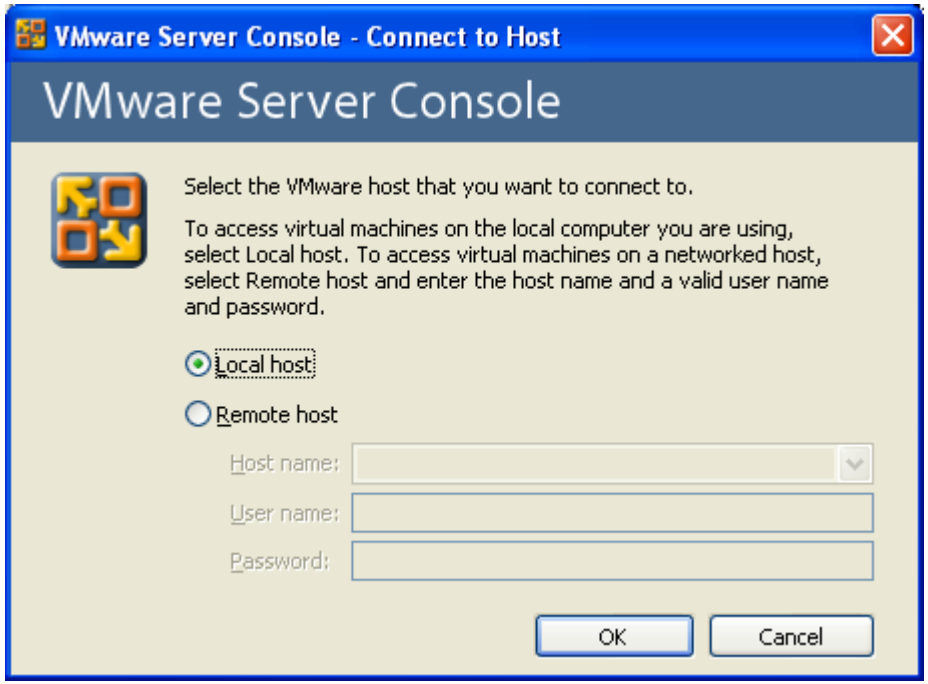


VMware Win Setup: /presentation /screenshots /01b_win_vmware_server_screenshots

01_VMware_Server_Console.PNG

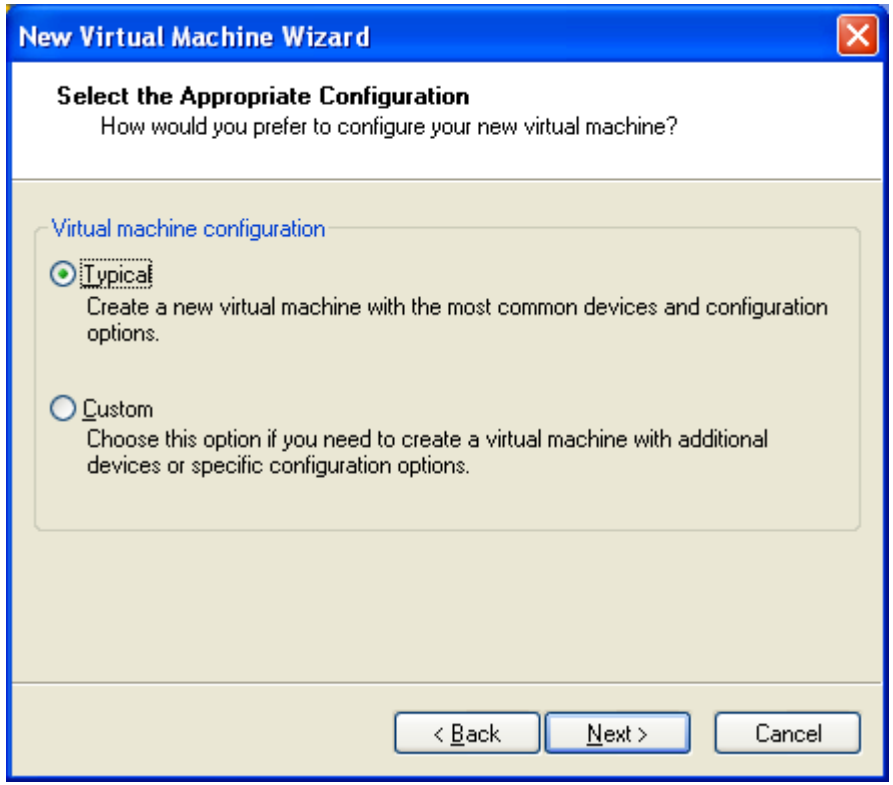


02_Connect_To_Host.PNG

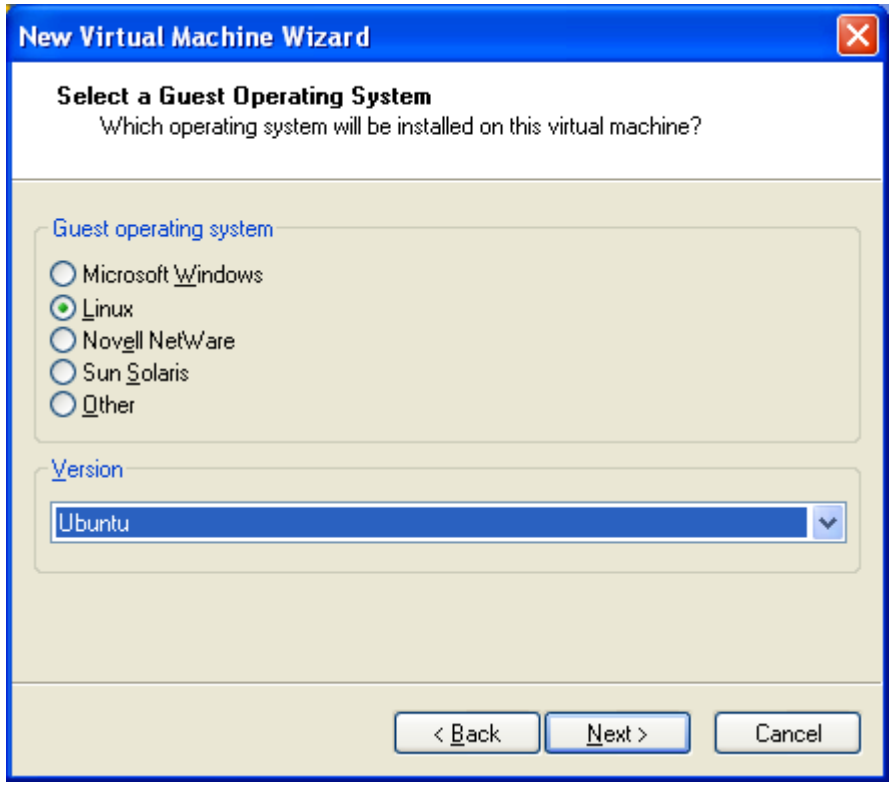


03_New_Virtual_Machine.PNG

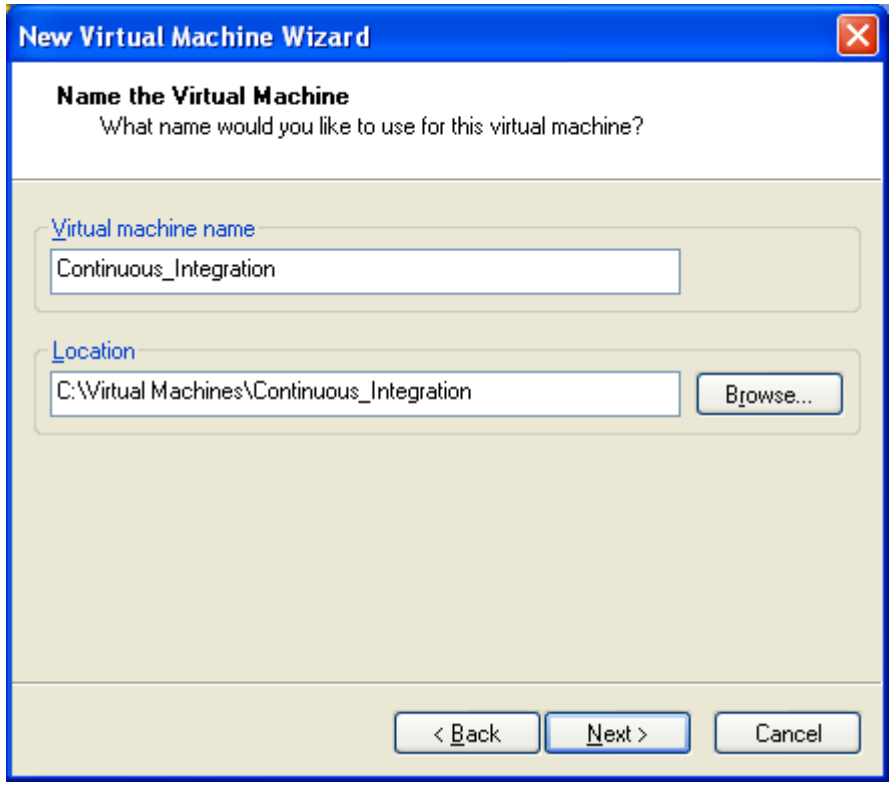
04_Virtual_Machine_Configuration.PNG



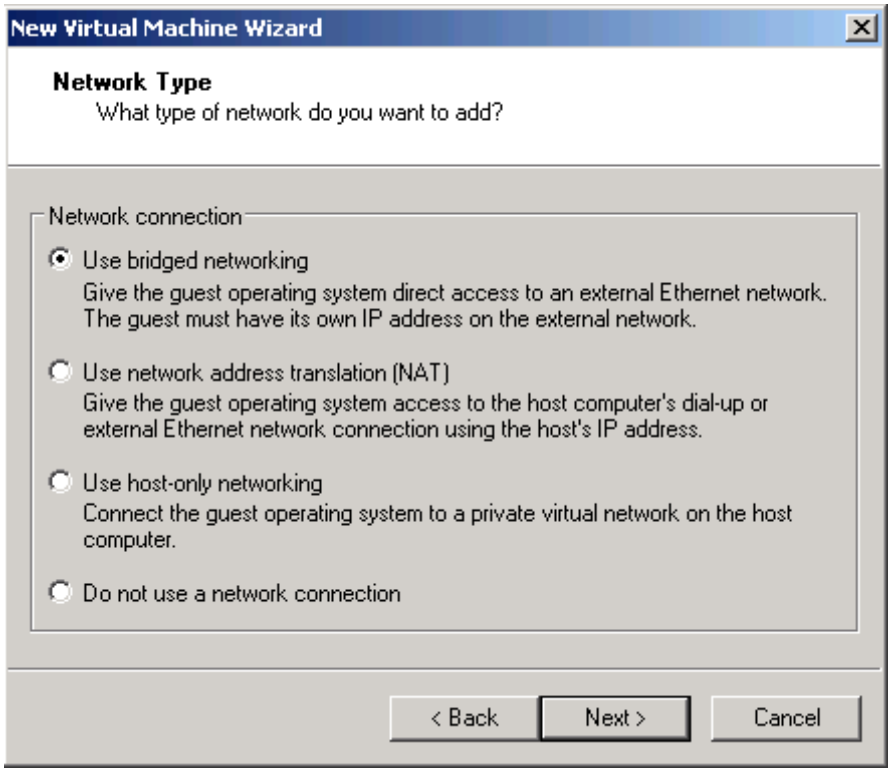
05_Select_a_Guest_Operating_System.PNG



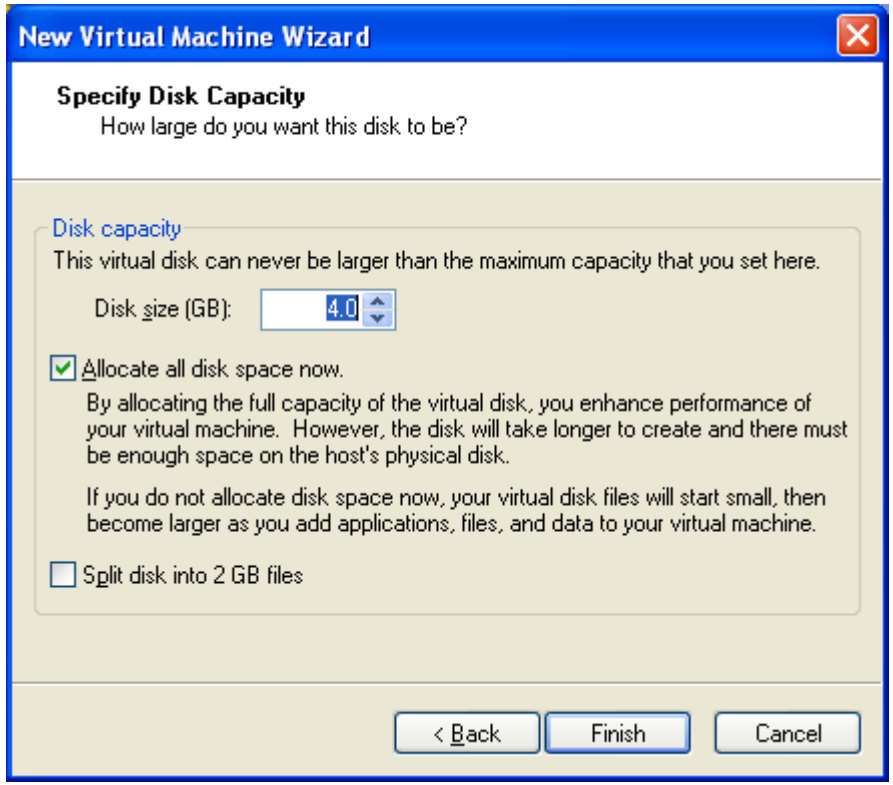
06_Name_the_Virtual_Machine.PNG



07_Network_Type.PNG

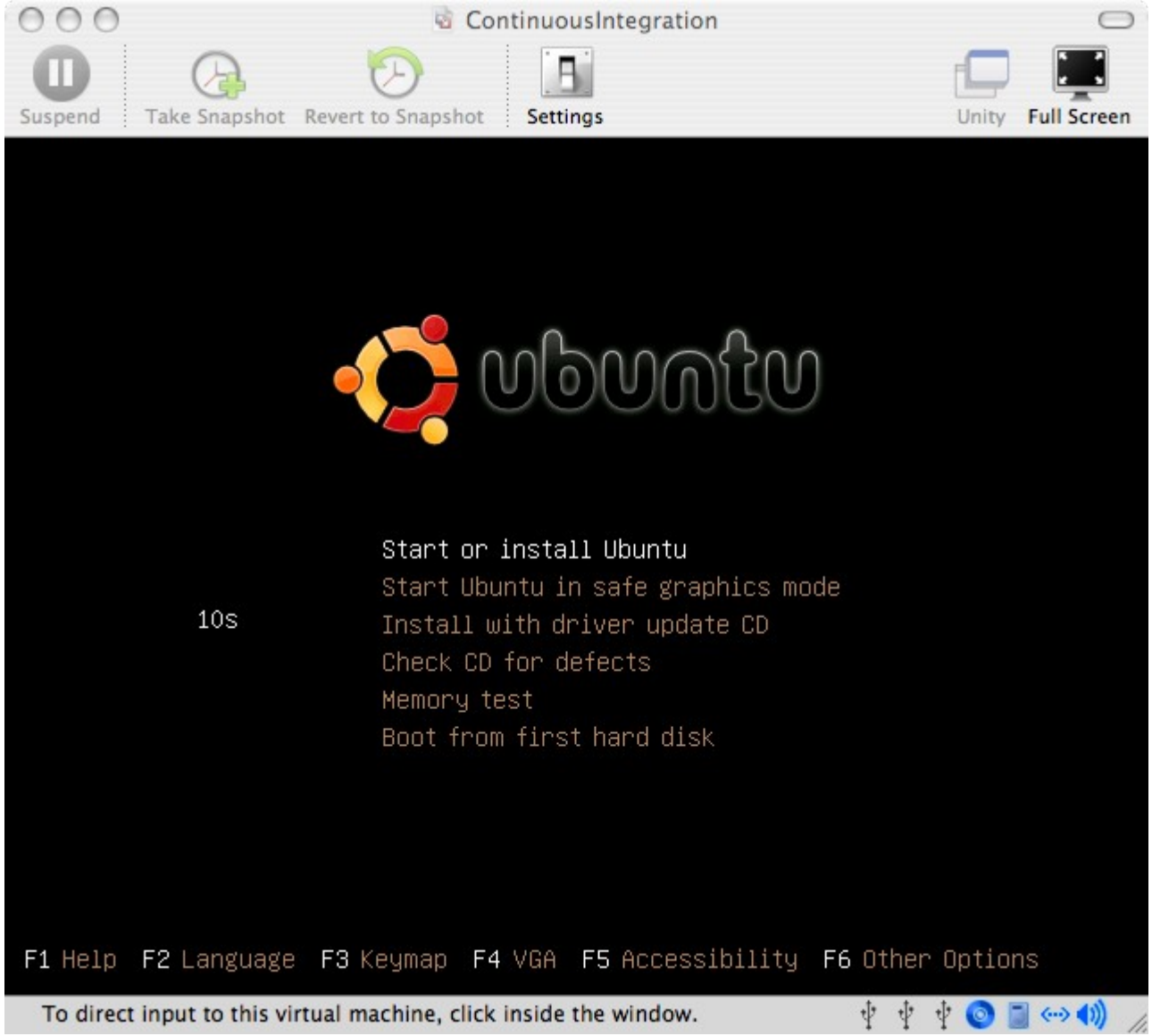


08_Specify_Disk_Capacity.PNG

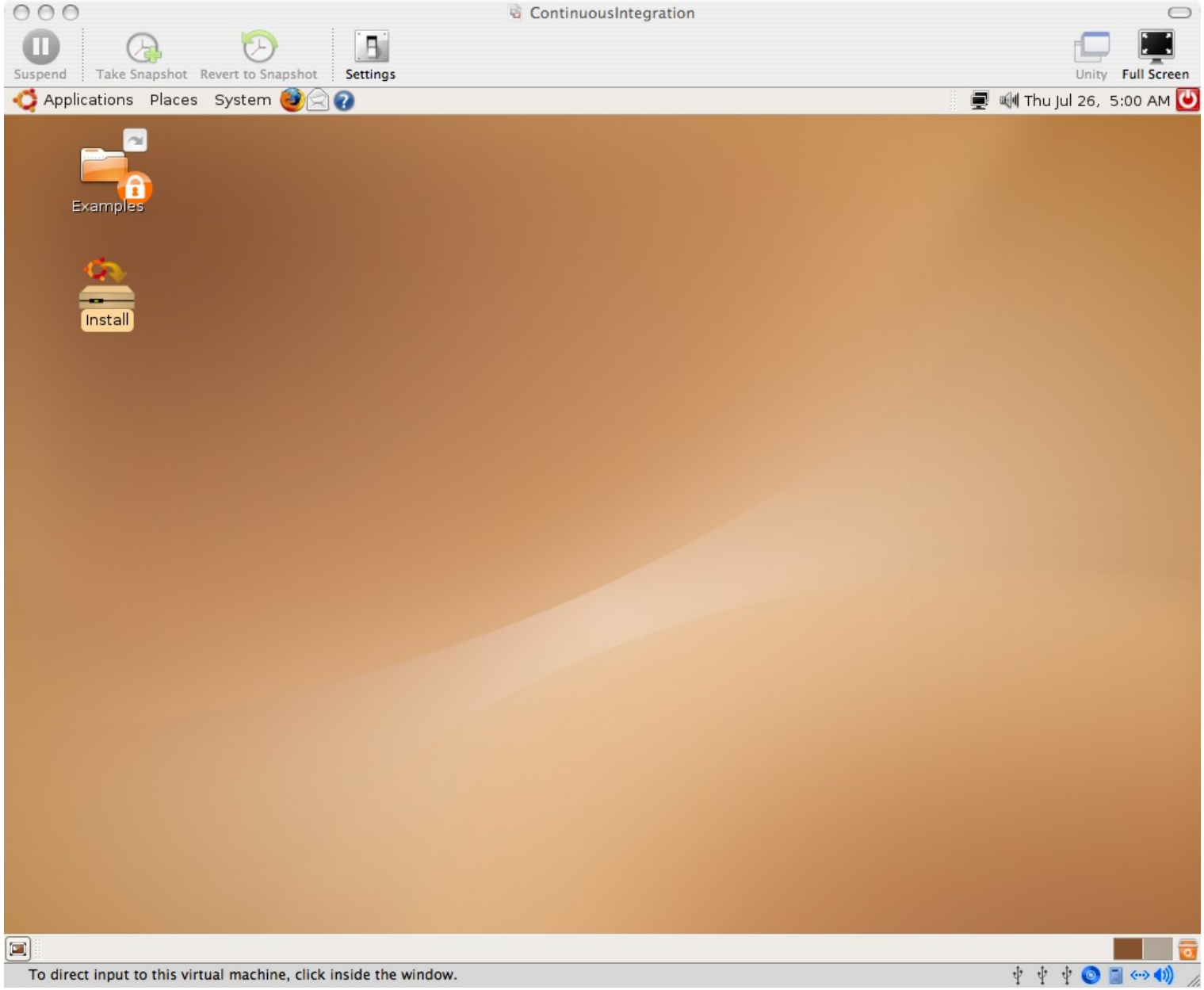


Mac/Win Ubuntu VM Setup: /presentation /screenshots /02_ubuntu_vm_ setup_screenshots

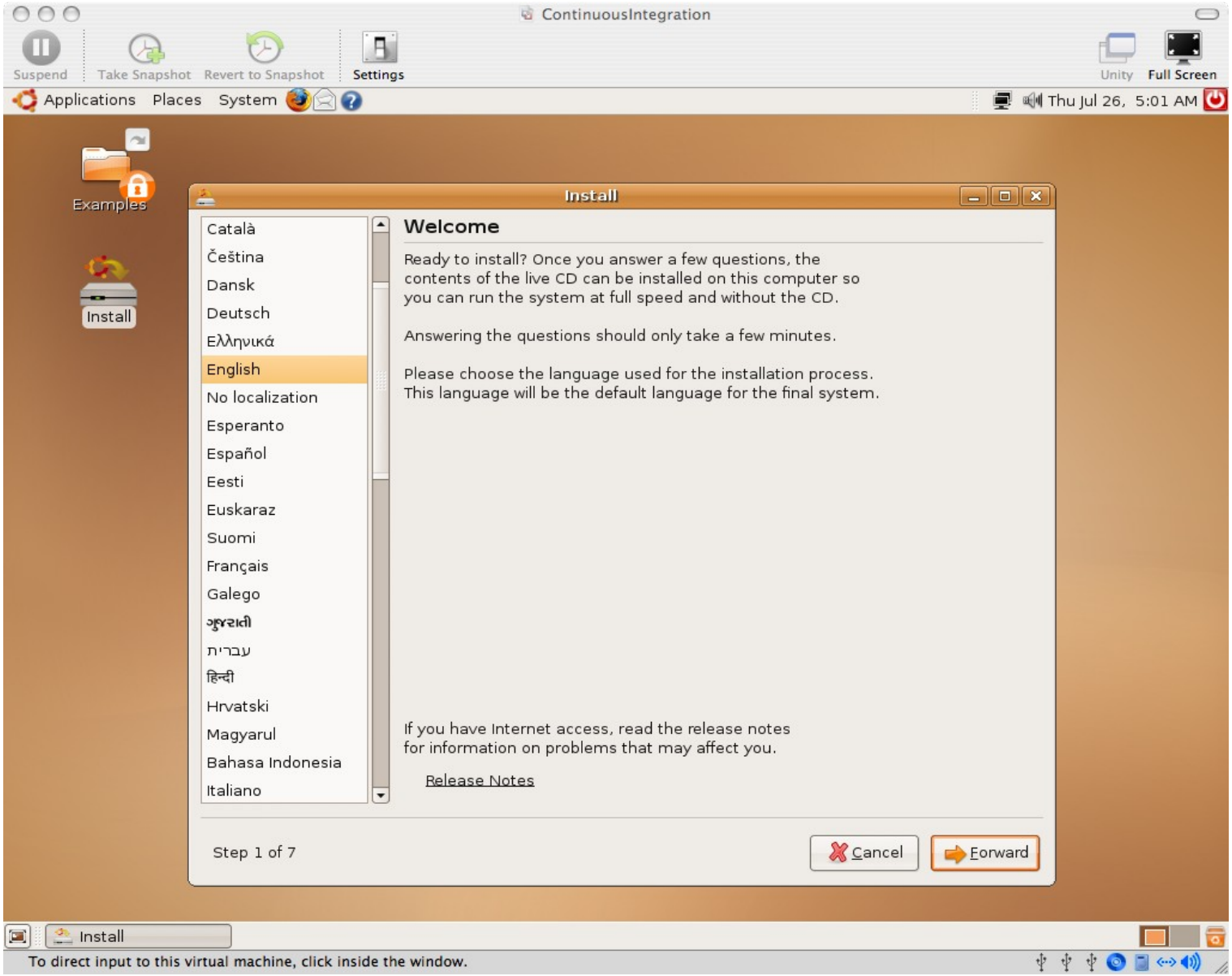
01_Start_or_Install_Ubuntu.png



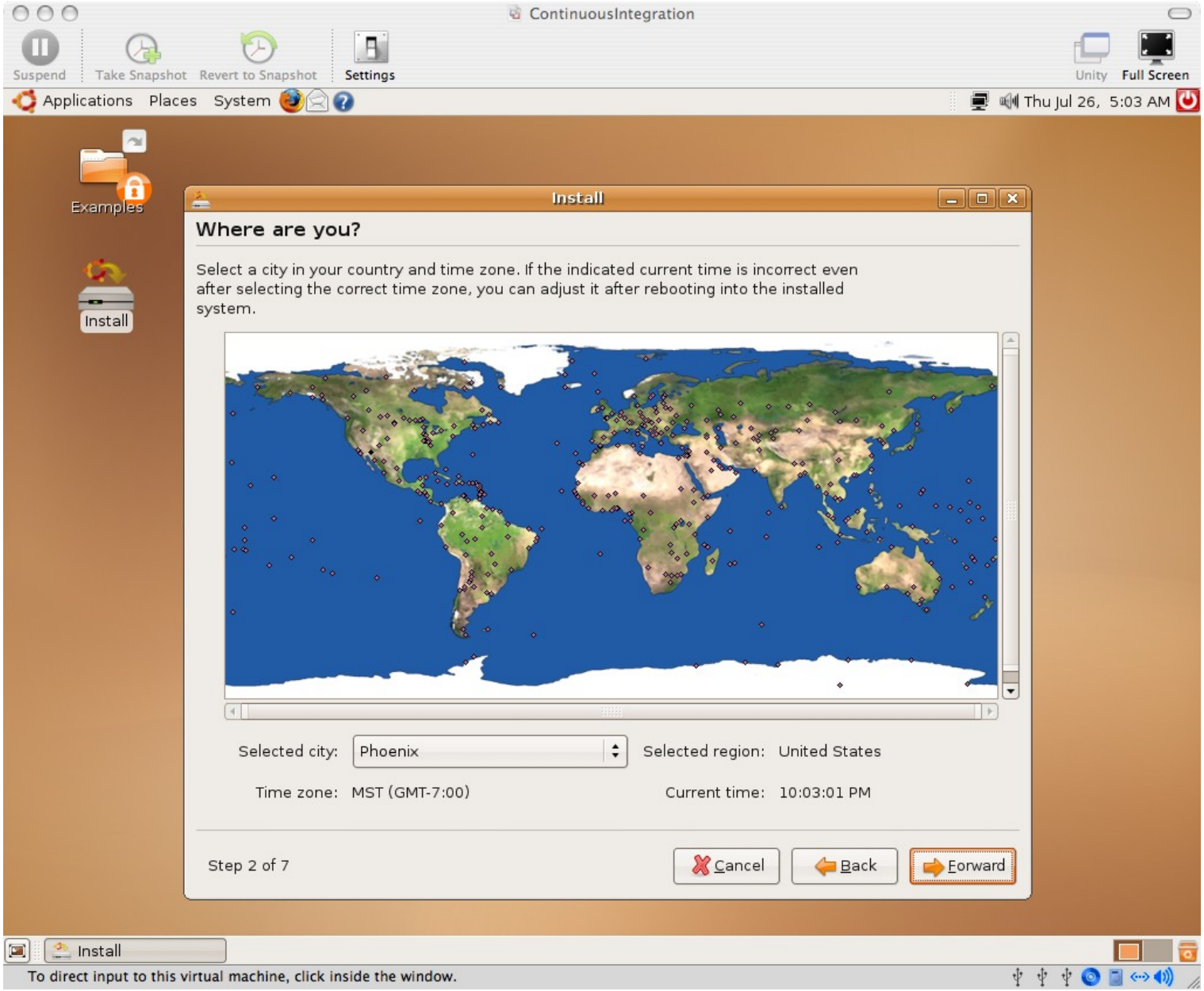
02_Install_Icon.png



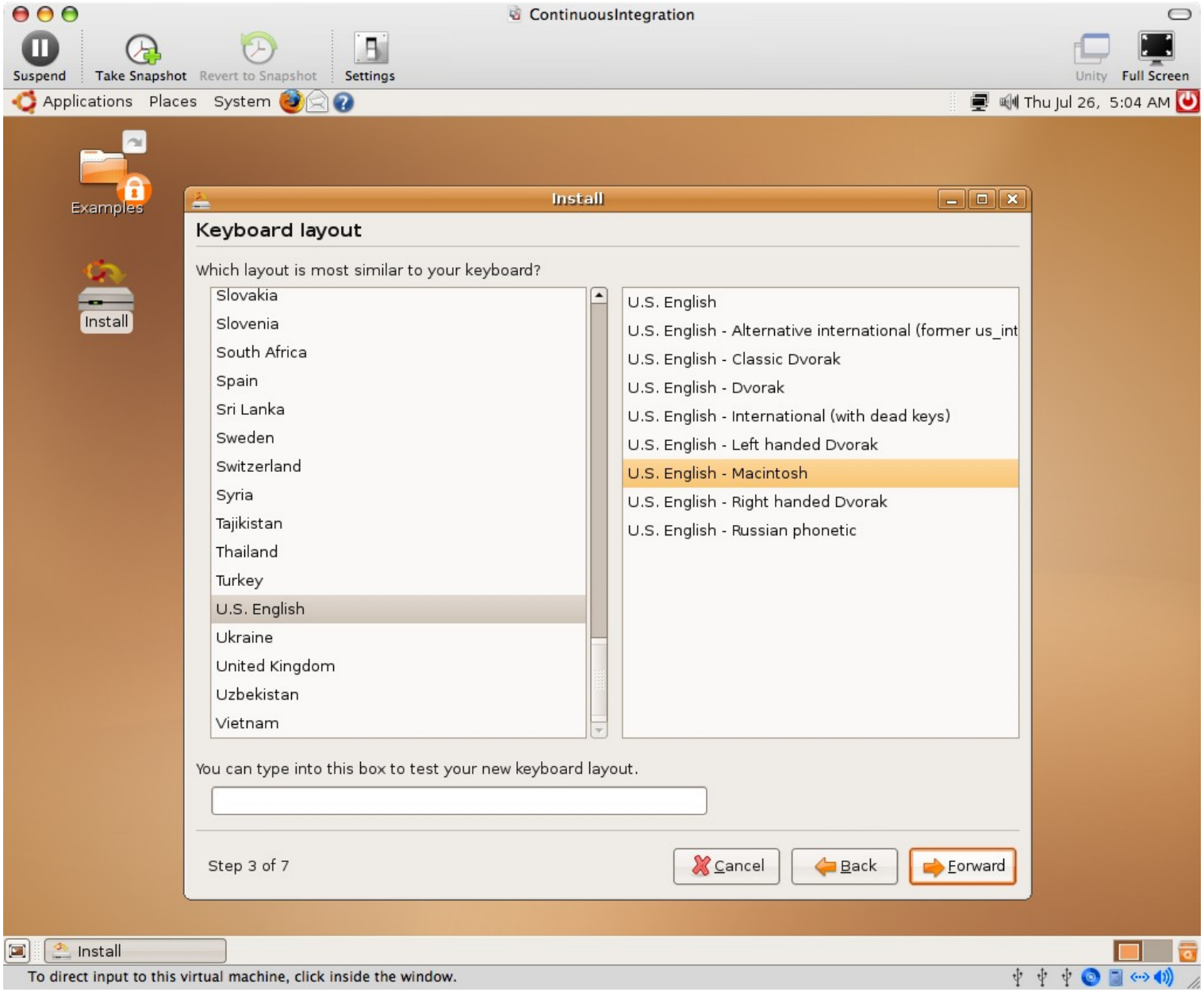
03_Welcome.png



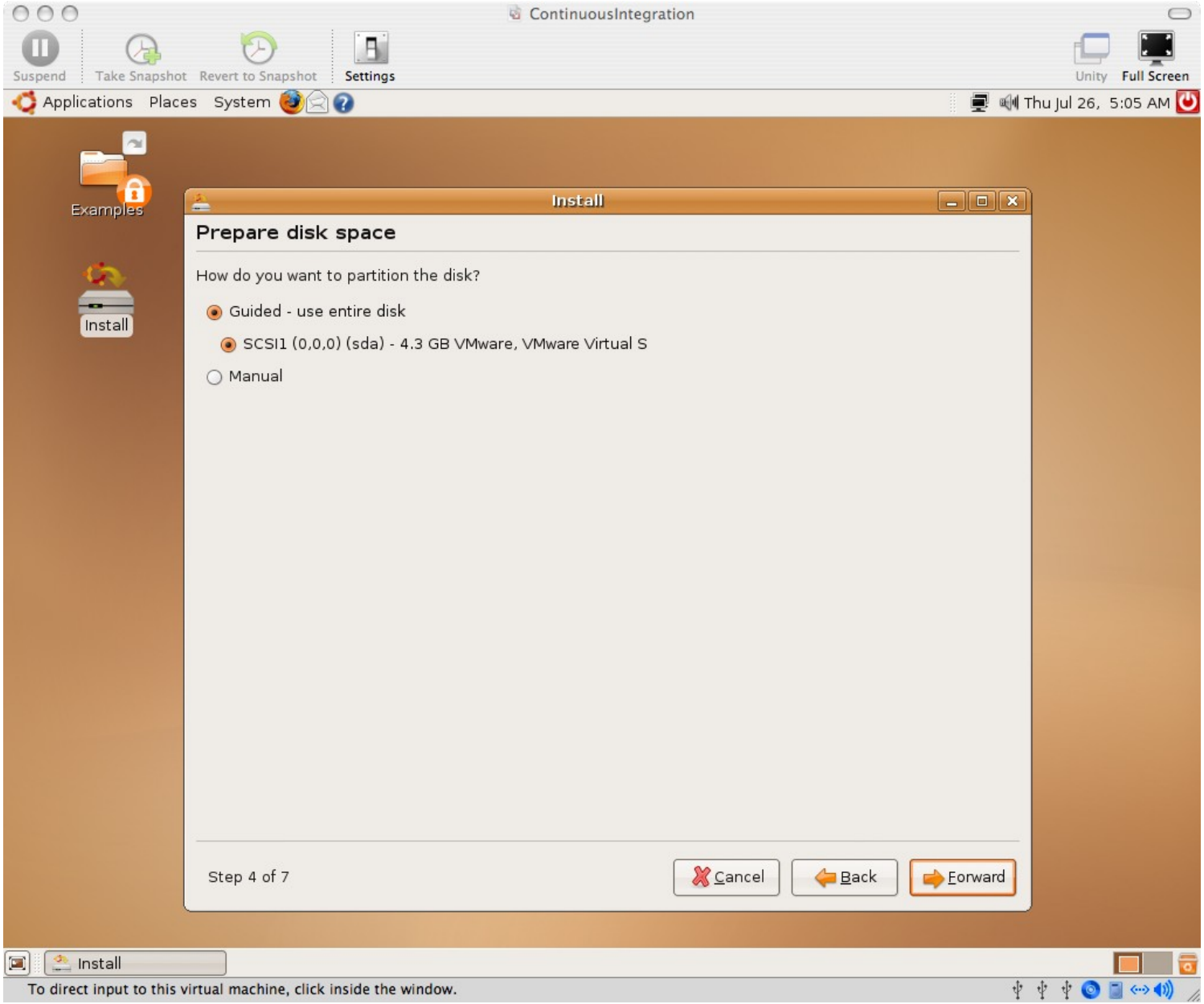
04_Where_are_you.png



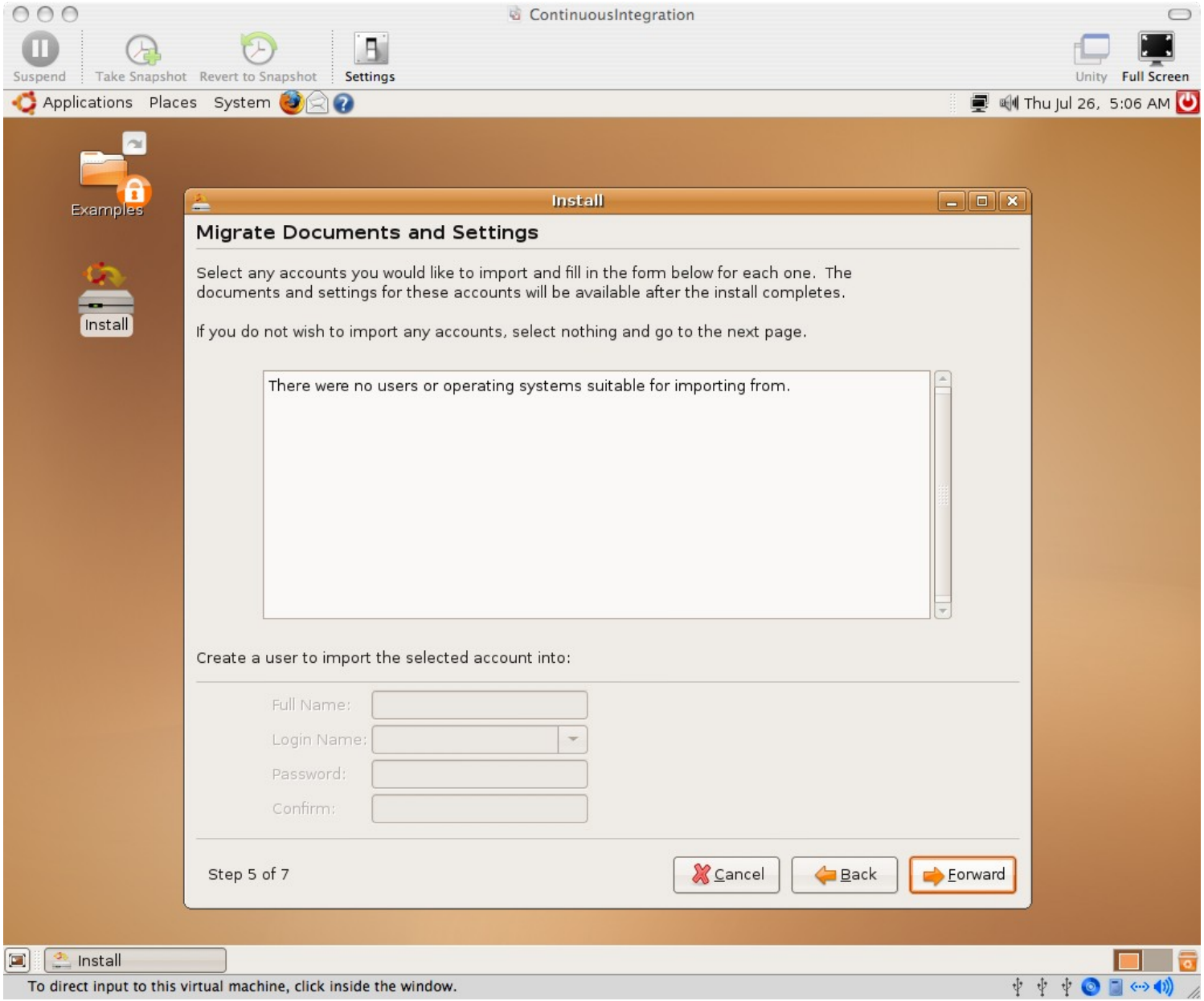
05_Keyboard_Layout.png



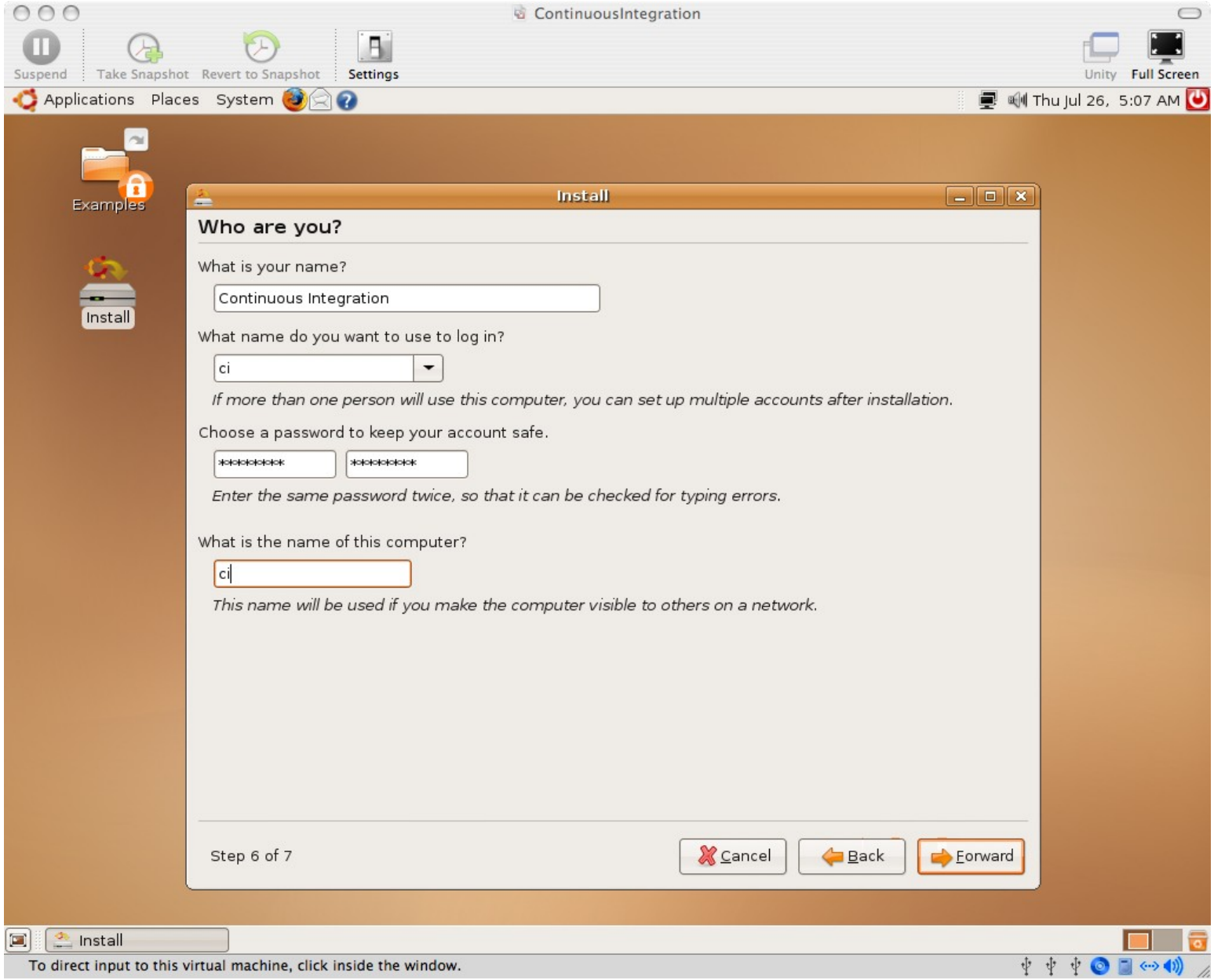
06_Prepare_disk_space.png



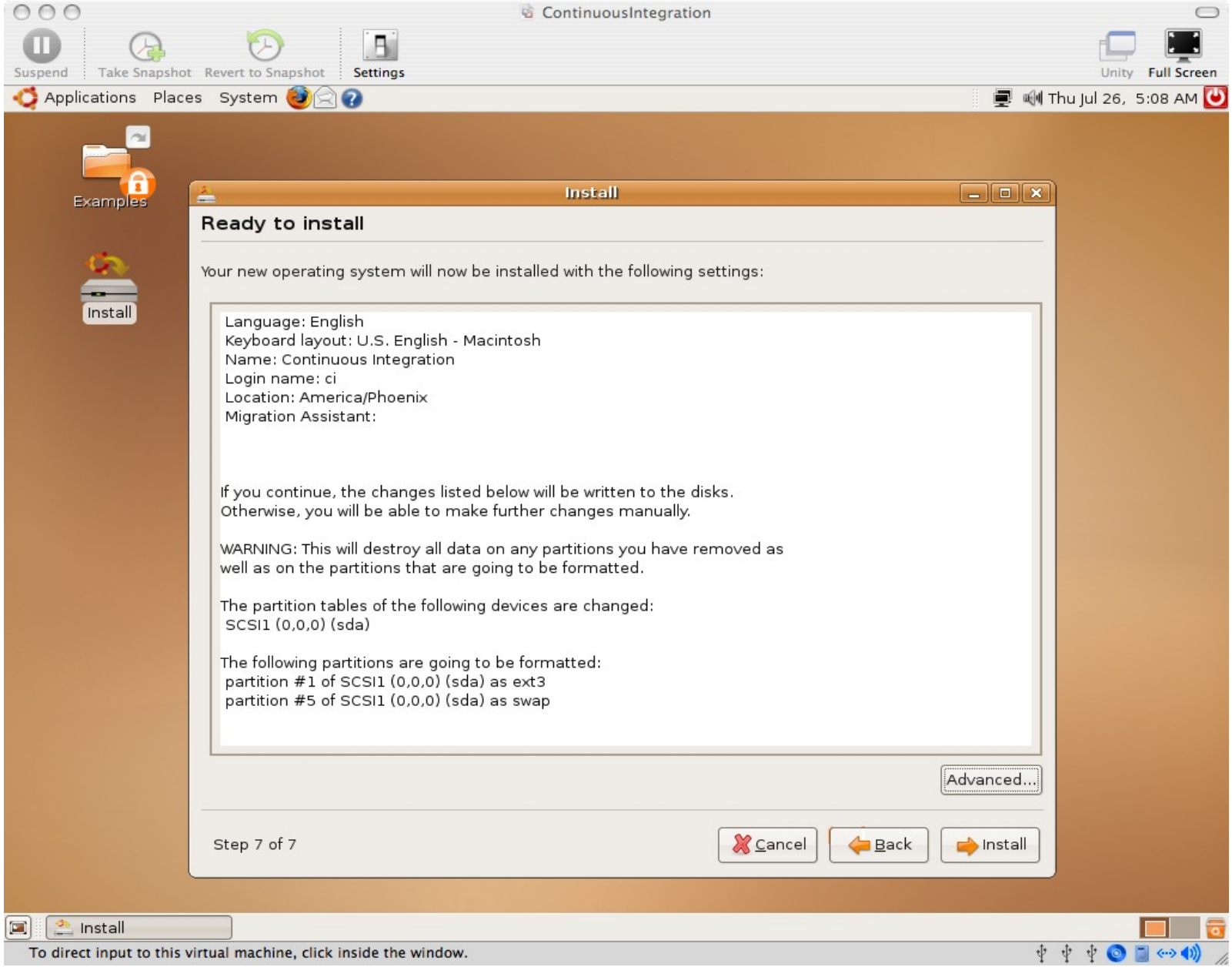
07_Migrate_Documents_and_Settings.png



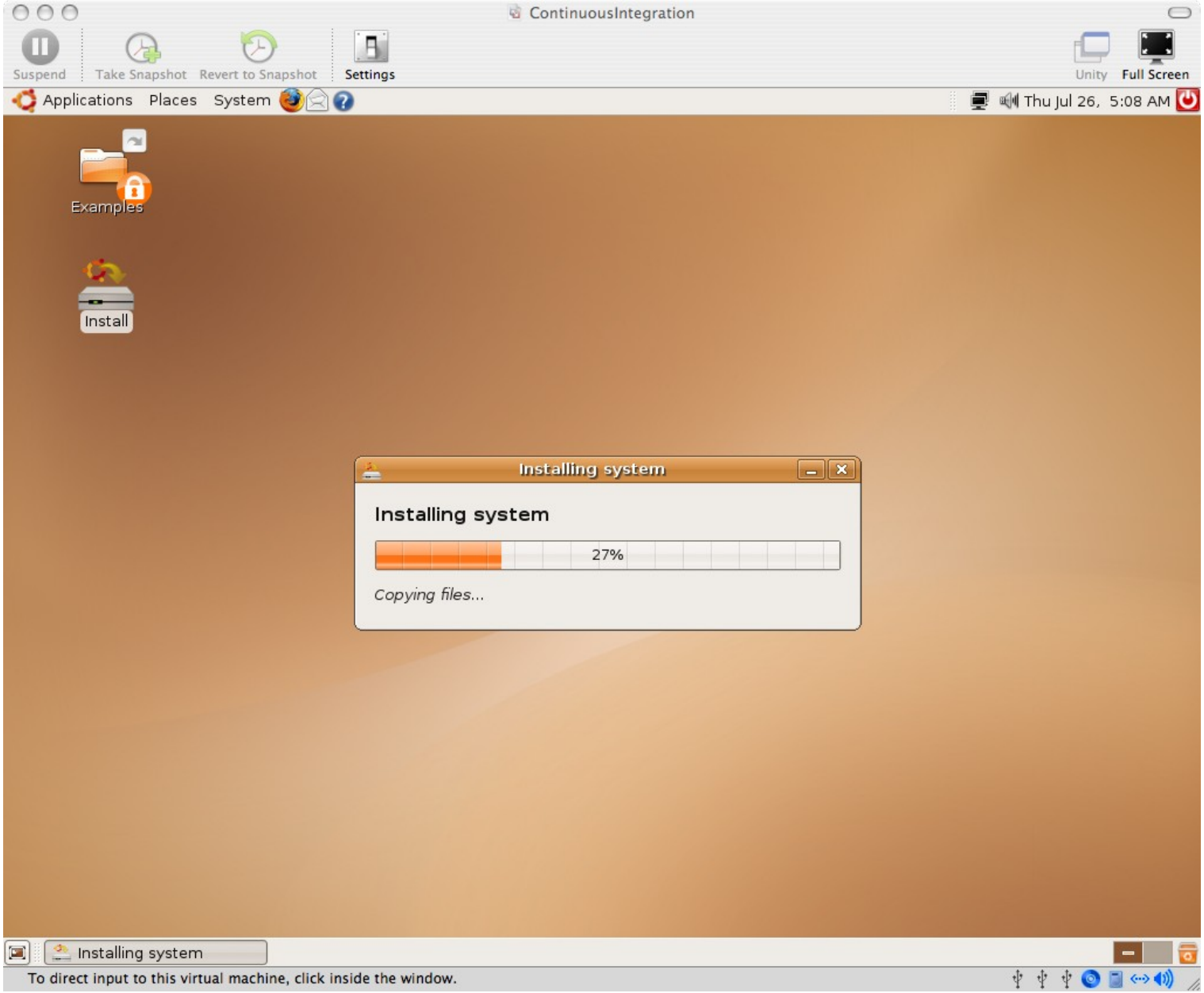
08_Who_are_you.png



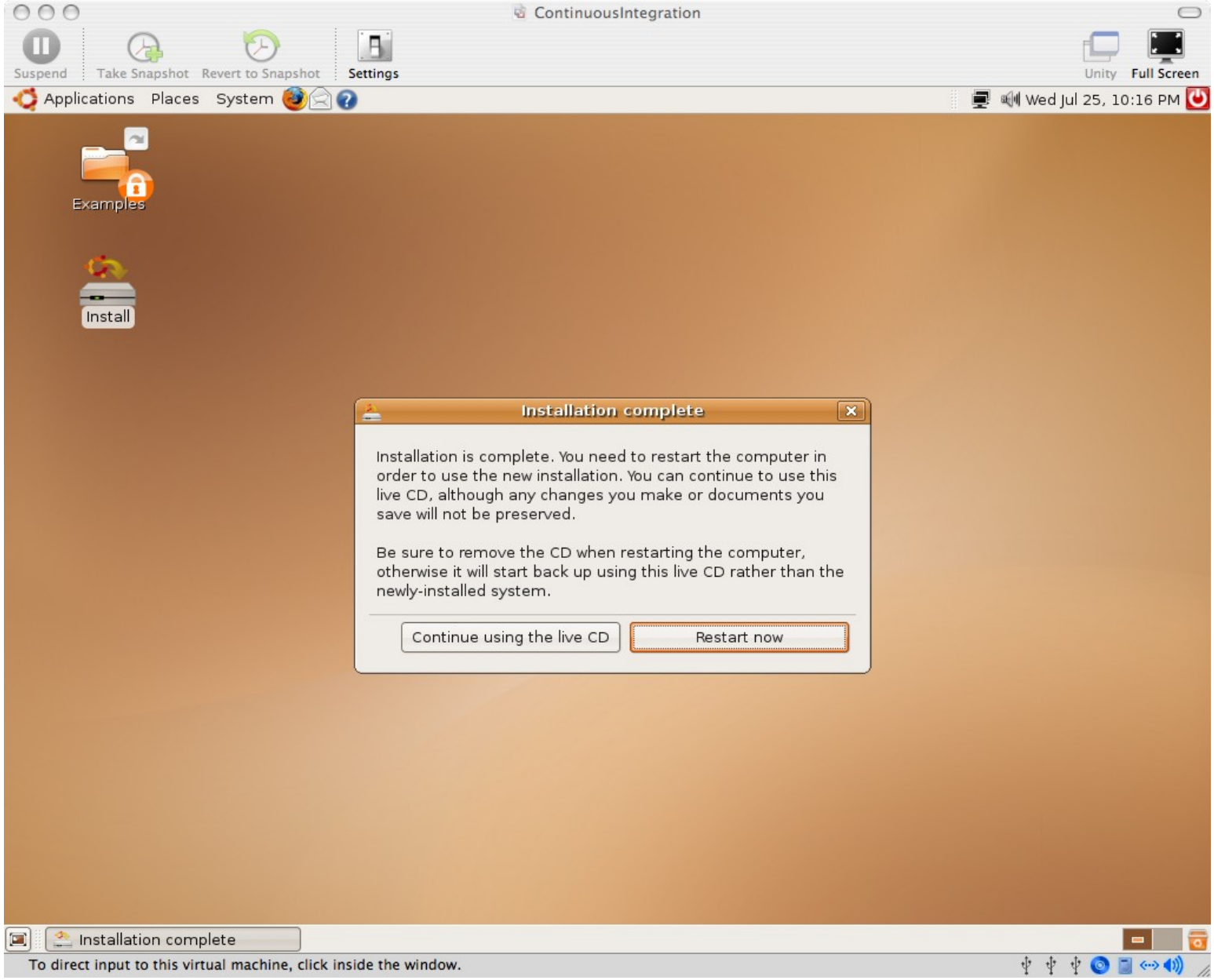
09_Ready_to_install.png



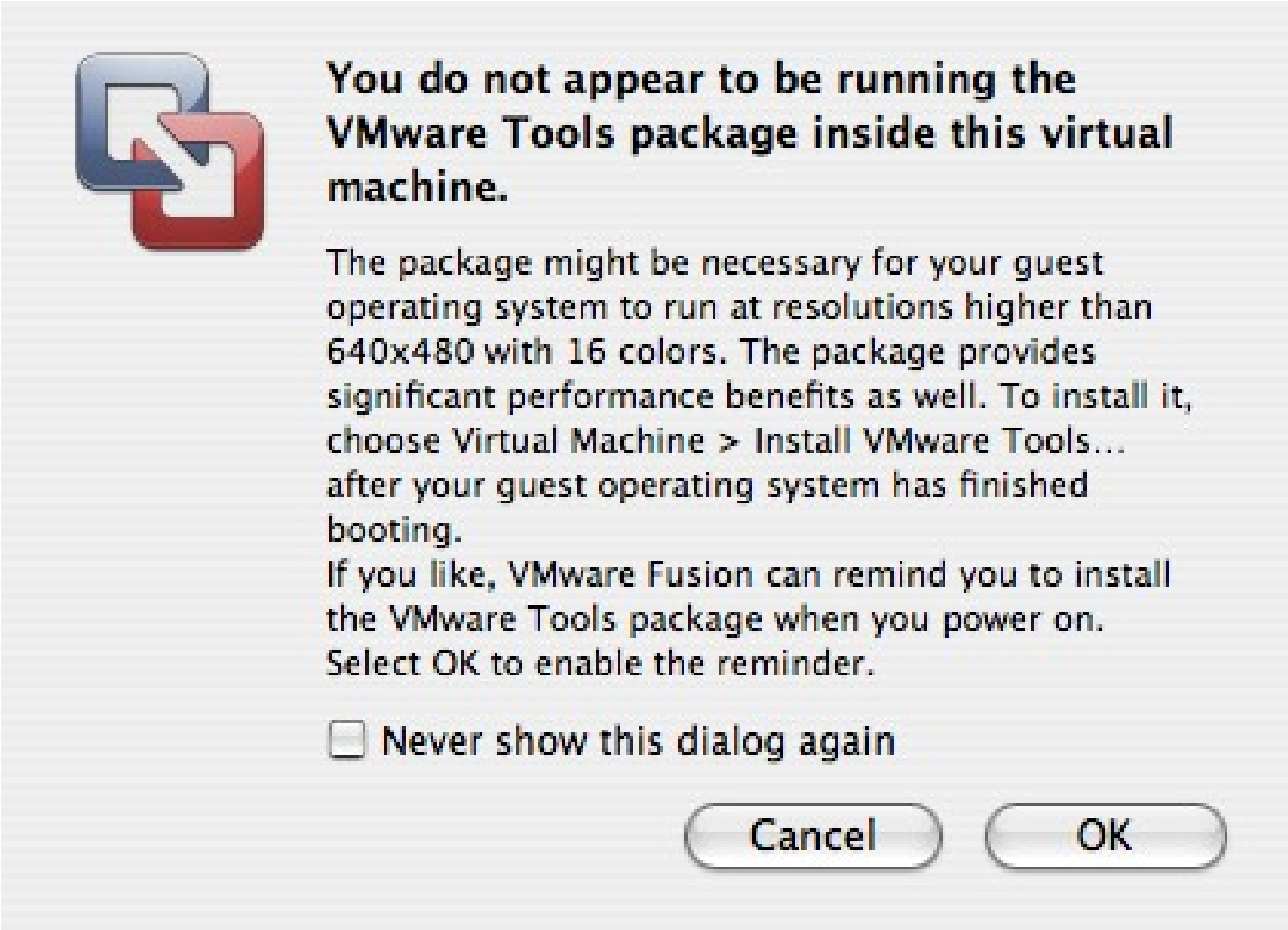
10_Installing_system.png



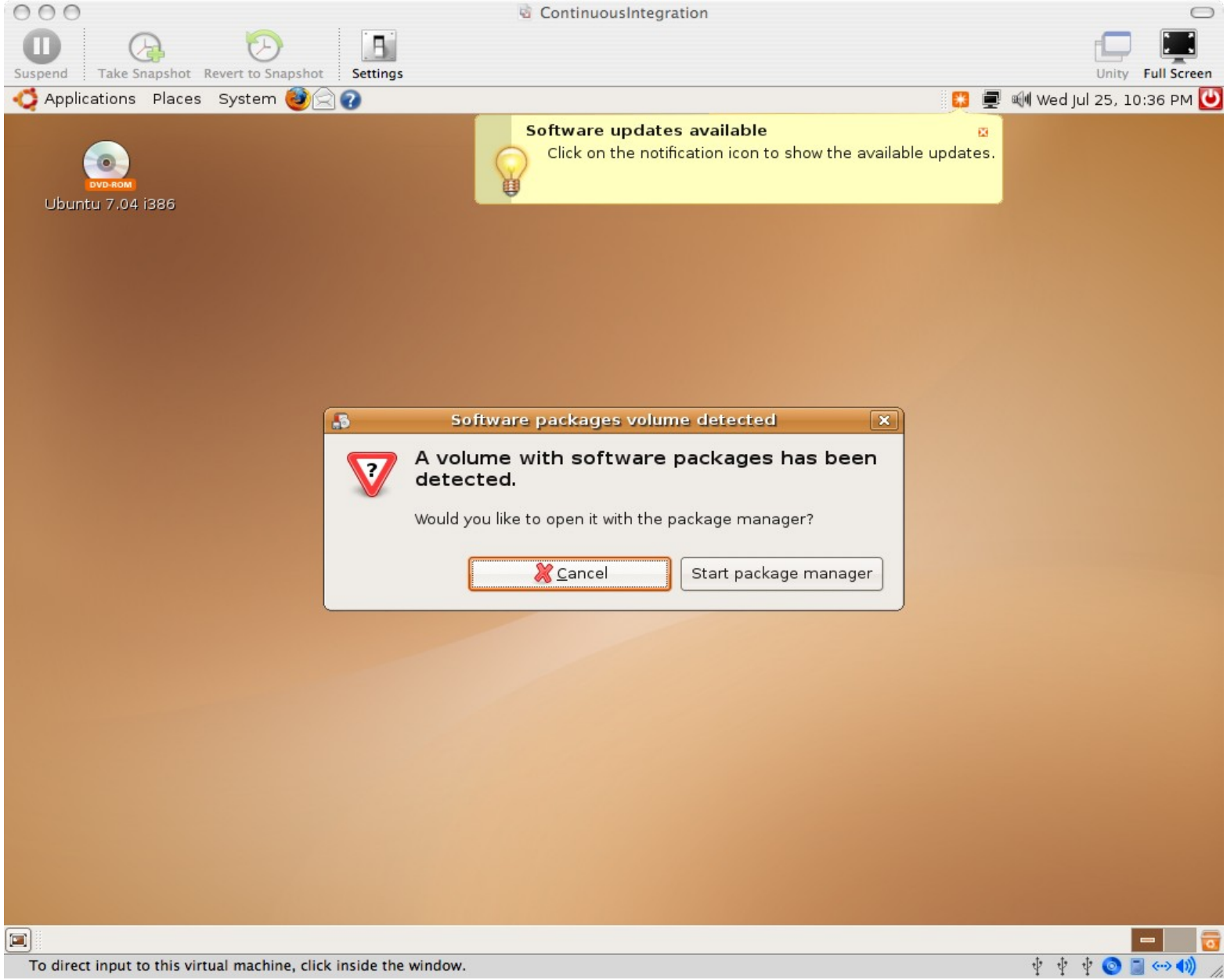
11_Installation_complete.png



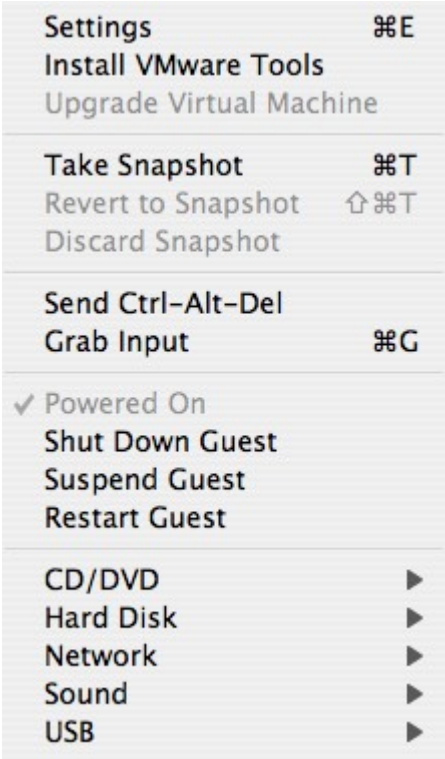
12_VMware_Tools_reminder.png



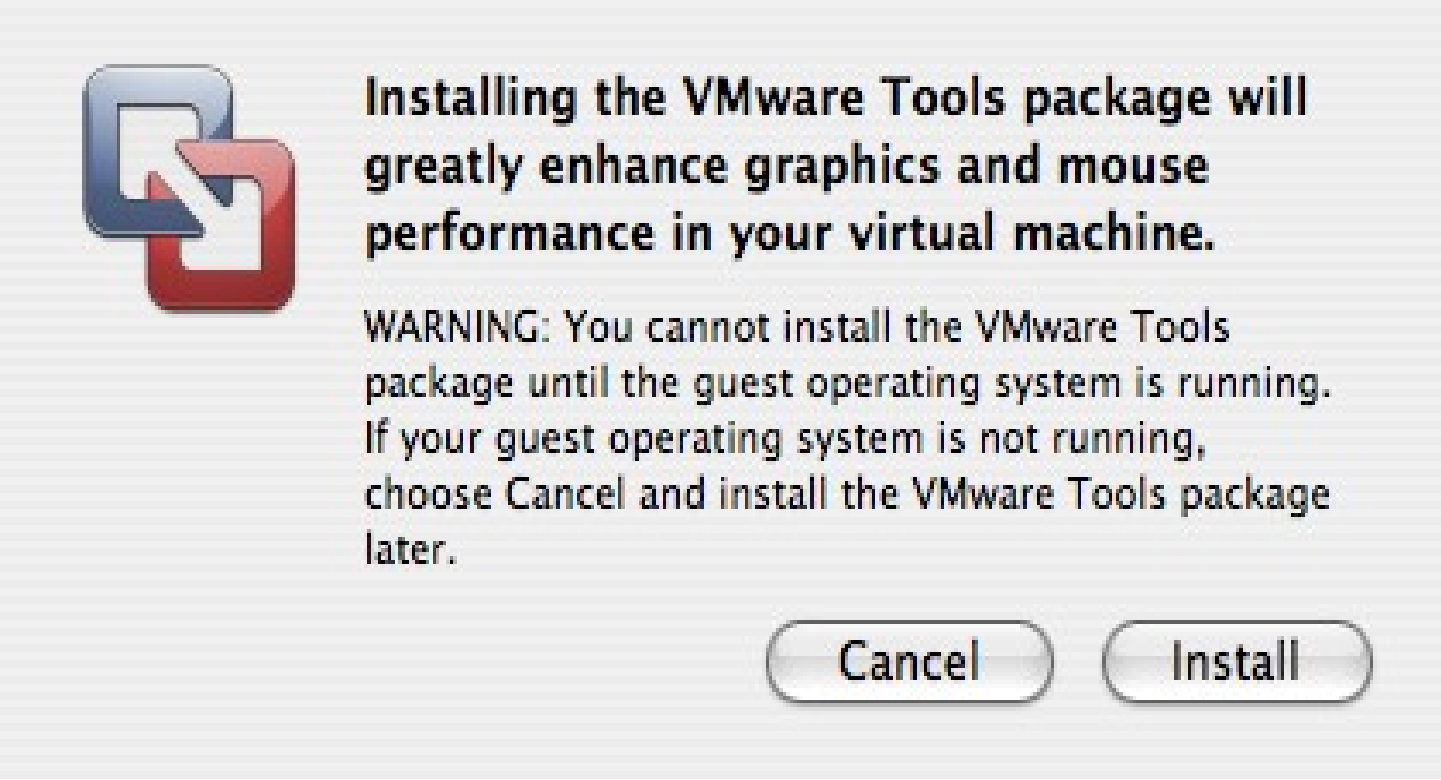
13_Cancel_package_manager.png



14_Virtual_Machine_Menu_Install_VMware_Tools.png

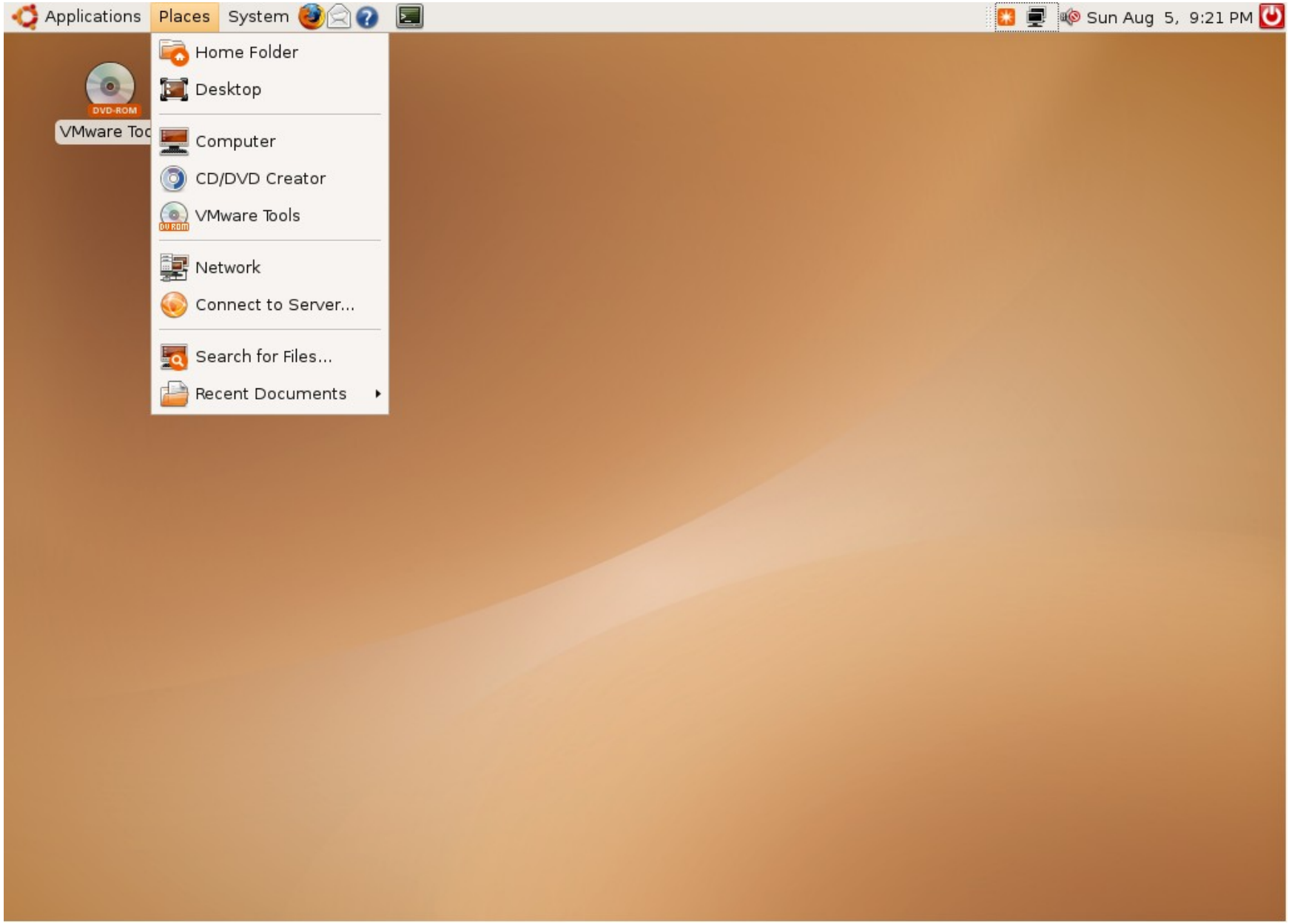


15_Installing_the_VMware_Tools_package.png

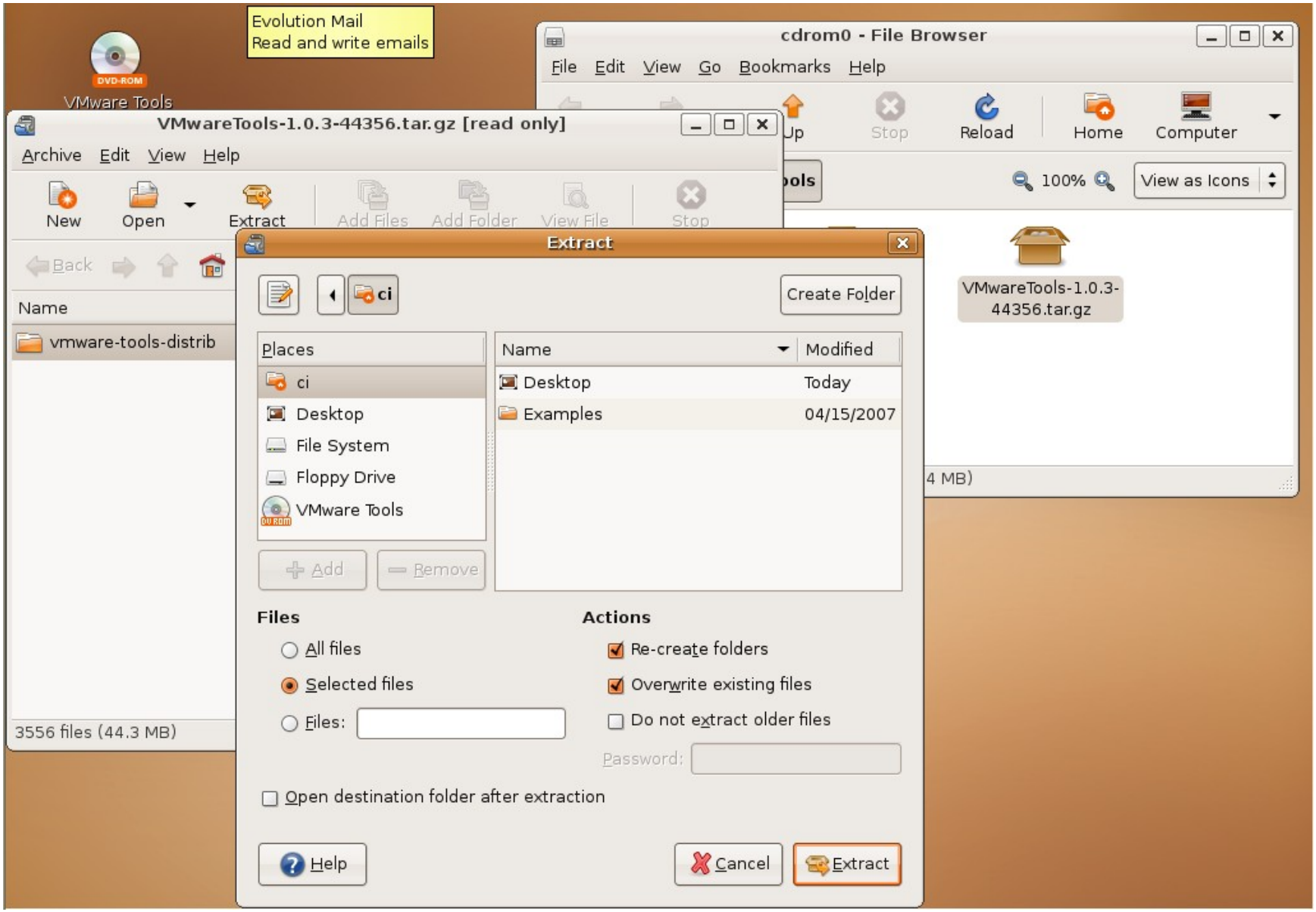


At this point, you may need to reboot (System -> Quit -> Restart) in order for the VMware Tools CD image to mount correctly, especially if you already have the Ubuntu ISO image mounted.

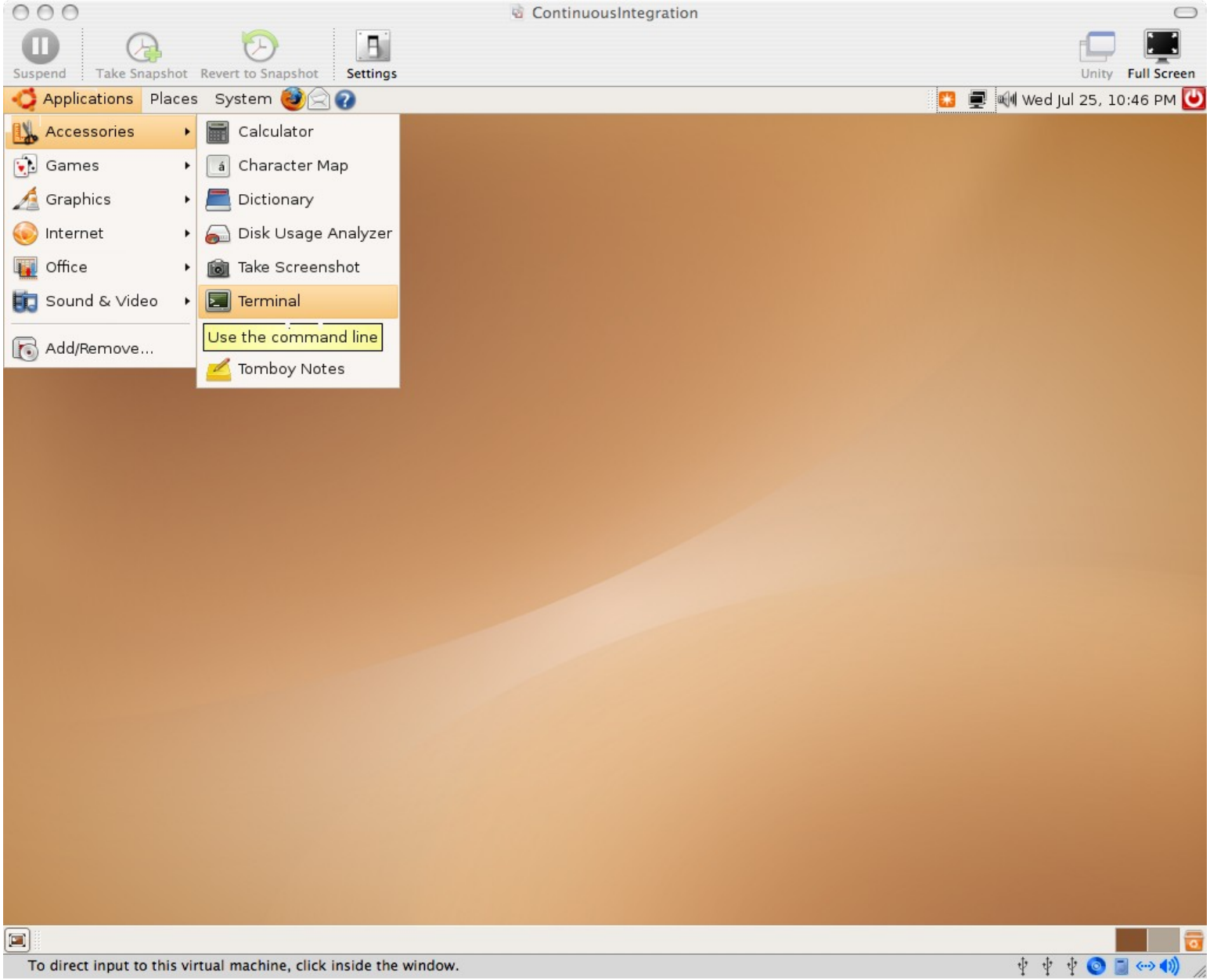
16_Open_VMWare_Tools_Image.png



17_Extract_VMware_Tools.png



18_Applications_Accessories_Terminal.png

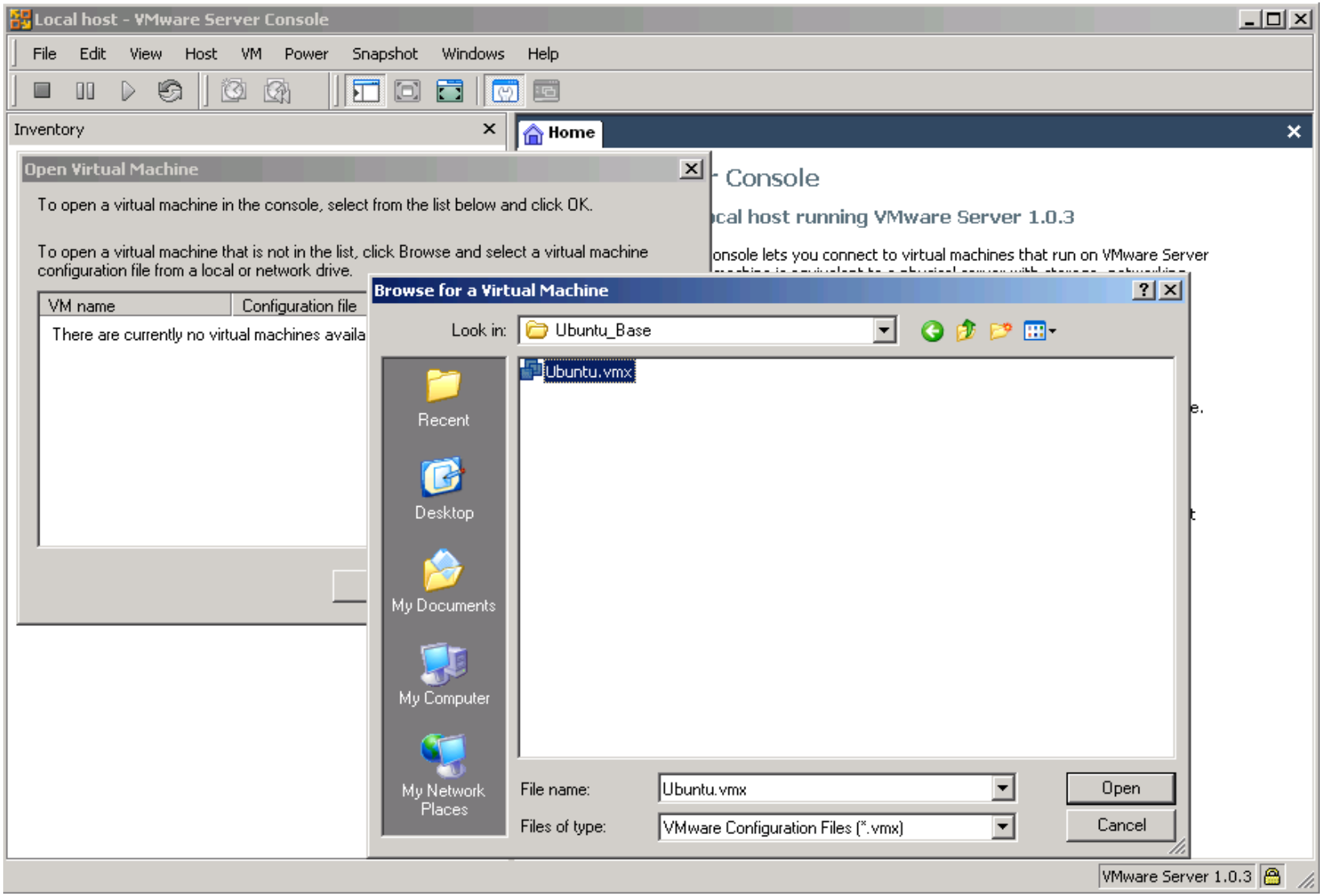


Install VMware Tools (Optional):

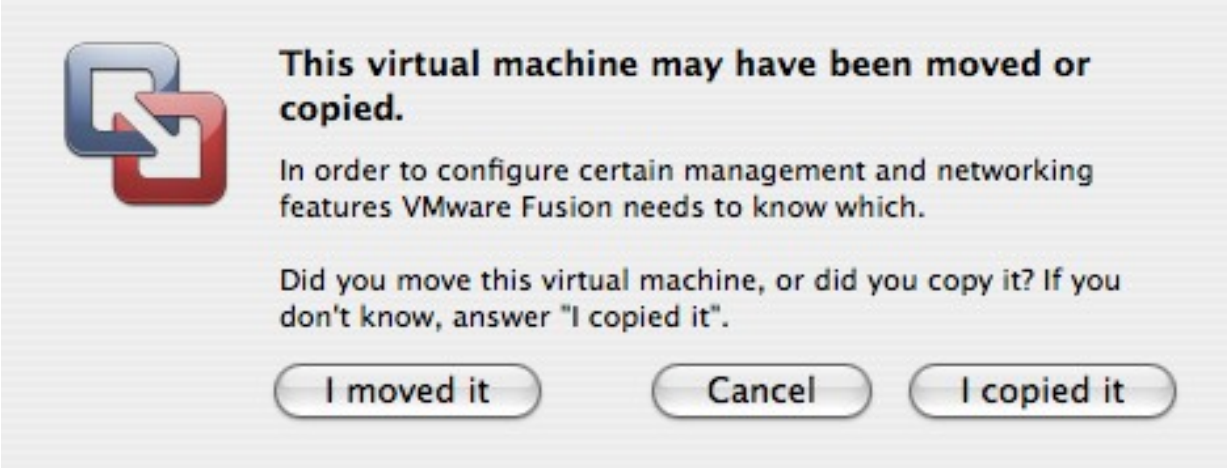
```
$ cd ~/vmware-tools-distrib
$ sudo ./vmware-install.pl
# enter password for sudo
# hit enter repeatedly to accept defaults for all prompts
# reboot (System -> Quit -> Restart)
```

Opening an existing VM Image Copy: /presentation /screenshots /03_virtual_machine_copy

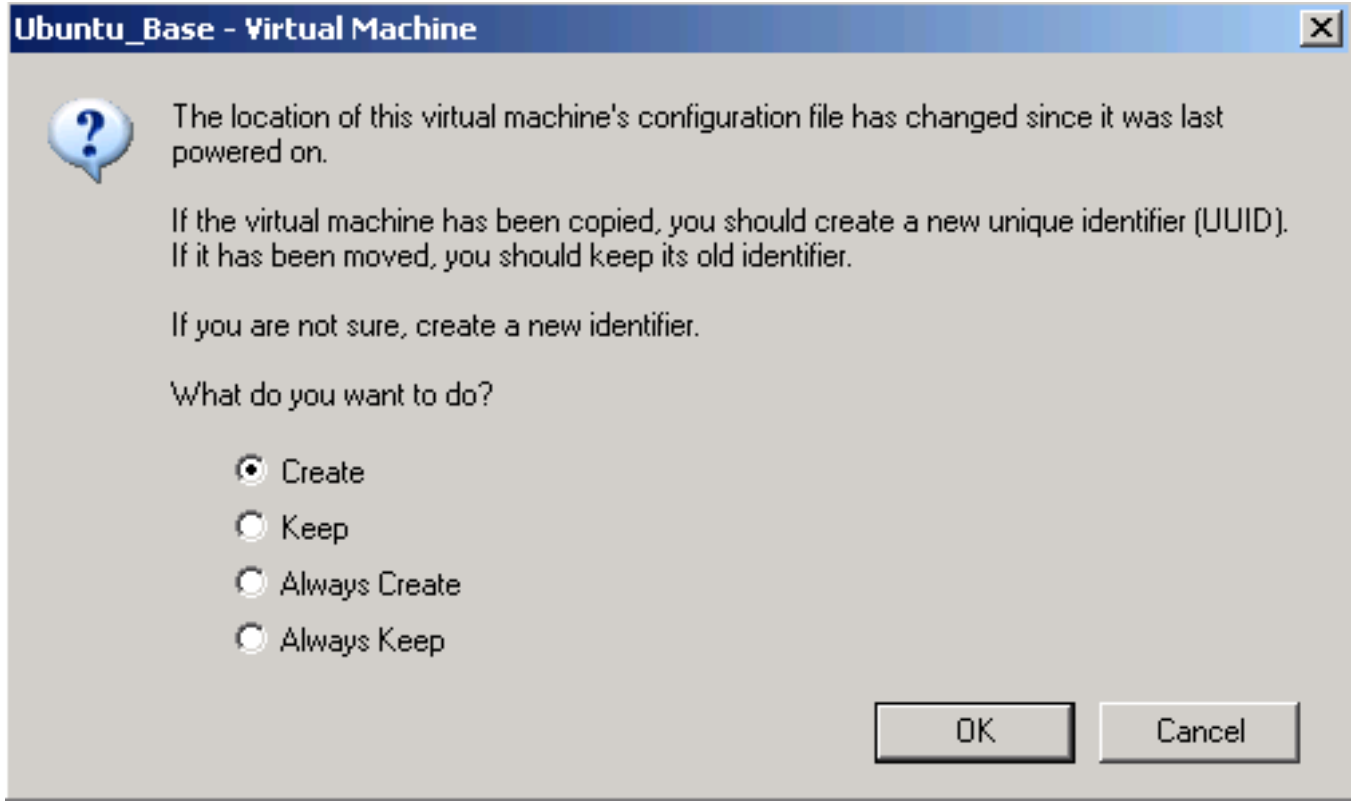
01_Browse_for_a_Virtual_Machine.PNG



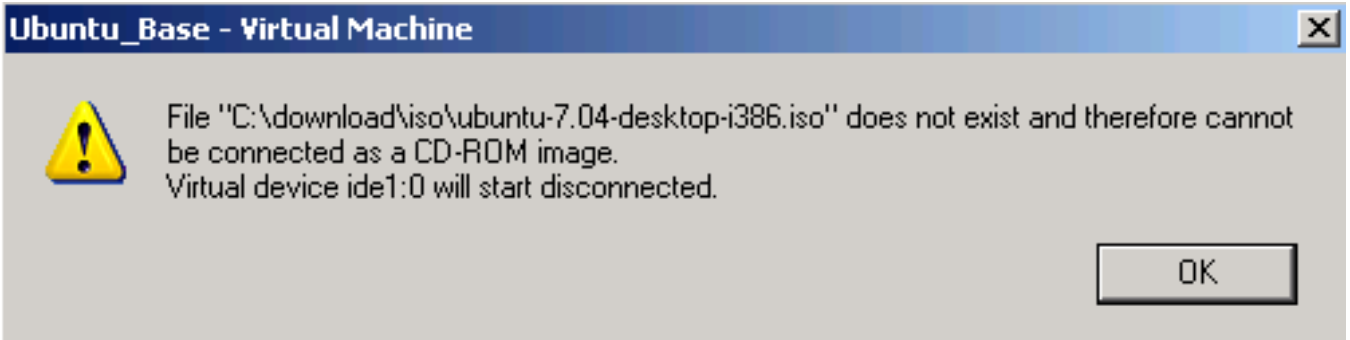
02a_Mac_Virtual_Machine_Copy.png



02b_Win_Virtual_Machine_Copy.png



03_Missing_ISO_CDROM_Image.PNG



Other Ubuntu Tweaks (Optional):

* System -> Preferences -> Screen Resolution
* System -> Preferences -> Mouse
* Drag Applications -> Accessories -> Terminal icon to quick launch area
* Terminal -> Edit -> Current Profile -> Scrolling -> Scrollback = 99999
* Ctrl +, Ctrl - in Terminal to change font size

B. Install Prerequisites

Legend
$ == shell input
# == comment or instructions
(nothing) == editor input or stdin

Example:
# sudo should prompt for a password unless you've sudo'd recently
$ sudo ls
password
should get file list

We will keep everything in the home dir, or "~" You can put it wherever you want

Ruby Packaging on Ubuntu/Debian: Plan9 vs FHS and LSB == confusing

You can install ruby via apt-get, but building it from source is recommended.

Install Ruby from source:

```
# install all prereqs/extensions in case you need them
$ sudo apt-get update
$ sudo apt-get install -y zlib1g zlib1g-dev
$ sudo apt-get install -y libssl-dev openssl
$ wget ftp://ftp.ruby-lang.org/pub/ruby/ruby-1.8.5.tar.gz
$ tar -zxvf ruby-1.8.5.tar.gz
$ cd ruby-1.8.5
$ gedit ext/Setup
# Uncomment all “non-Win” lines (all except Win32API and win32ole) by removing “#”
$ ./configure
$ make
$ sudo make install
```

Install RubyGems:

```
$ wget
http://rubyforge.org/frs/download.php/20989/rubygem
s-0.9.4.tgz
$ tar -zxvf rubygems-0.9.4.tgz
$ cd rubygems-0.9.4
$ sudo ruby setup.rb
```

Install Sun java:

```
$ sudo apt-get install -y sun-java5-bin
# accept all prompts
```

Install MySql (required by default Rails app):
$ sudo apt-get install -y mysql-server

Install subversion:
$ sudo apt-get install -y subversion

Install ant:

```
$ sudo apt-get install -y ant
$ sudo apt-get install -y ant-optional
# By default, this uses Gnu java, not Sun's...
```

```
Install mozilla as an alternate browser
# because jsunit will kill the browser it is testing
# libgtk1.2 is a dependency
$ sudo apt-get install -y libgtk1.2
$ wget http://ftp-
mozilla.netscape.com/pub/mozilla.org/mozilla/release
s/mozilla1.7.13/mozilla-i686-pc-linux-gnu-1.7.13-
installer.tar.gz
$ tar -zxvf mozilla-i686-pc-linux-gnu-1.7.13-
installer.tar.gz
$ sudo mozilla-installer/mozilla-installer
# install Navigator only
$ /usr/local/mozilla/mozilla &
```

Create Subversion Repo

```
$ svnadmin create repo
```

C. Create sample Ruby on Rails Project

Install Rails

```
$ sudo gem install rails --include-dependencies
```

Create a rails project

```
$ rails mysite
$ cd mysite
```

Create databases for rails project

```
$ mysql -u root
mysql> create database mysite_development;
mysql> create database mysite_test;
mysql> create database mysite_production;
# (prod needed because cruise complained if it was
not there)
mysql> exit
```

Hack rails database.yml to match debian defaults

```
$ gedit config/database.yml
# add the following entry to all three databases
socket: /var/run/mysqld/mysqld.sock
# NOTE: Sometimes, Rails will do this for you
automatically...
```

Create a rails migration and db table

```
$ ruby script/generate migration CreateUserTable
$ gedit db/migrate/001_create_user_table.rb
  def self.up
    create_table "users" do |t|
      t.column "name", :string
    end
  end

  def self.down
    drop_table "users"
  end
$ rake db:migrate
```

Remove default index.html and create a page

```
$ rm public/index.html
$ ruby script/generate scaffold User
$ gedit test/functional/users_controller_test.rb
# change "users(:first)" to "users(:one)" – there should only be one occurrence
```

Test rails site

```
$ rake  # should pass all tests
$ ruby script/server
# New Terminal Tab: File -> Open Tab or Ctrl-Shift-T
# should be in mysite dir
$ firefox http://localhost:3000/users
# create a user
```

Import site into subversion

```
# change back to home dir (~)
$ cd
# remove temp files we don't want to check in
$ rm -rf mysite/log/*
$ rm mysite/db/schema.rb
$ rm -rf mysite/tmp
$ svn import mysite file:///home/ci/repo/mysite -m
"import"
$ rm -rf mysite
$ svn co file:///home/ci/repo/mysite mysite
```

Set svn:ignores

```
# ignore all temp files, to have a clean workspace
$ cd mysite
$ export EDITOR=gedit
$ svn propedit svn:ignore .
tmp
logs
$ svn propedit svn:ignore log
*
$ svn propedit svn:ignore db
schema.rb
$ svn commit -m "ignores"
$ cd
```

D. cruisecontrol.rb setup

cruisecontrol.rb is still new. We will use a recent build, which has many features not found in the 1.1.0 release

Check http://cruisecontrolrb.thoughtworks.com/projects for a recent, successfully building revision. We'll use rev 521

Check out a recent build of CruiseControl.rb

```
$ svn checkout
http://cruisecontrolrb.rubyforge.org/svn/trunk/@521
cc
```

Do a temporary hack to fix a bug in cc.rb rev 521

```
$ cd cc
$ mkdir projects
$ echo '1' > projects/data.version
$ cd
```

Set up project in cruisecontrol

```
$ cd cc
$ ./cruise add MySite --url file:///home/ci/repo/mysite
$ ./cruise start
```

View cruisecontrol web page

```
# Ctrl-Shift-T for new Terminal tab
$ firefox localhost:3333
# click MySite
# Should be passing
```

Take this opportunity to familiarize yourself with cruisecontrol.rb. It's not covered here ;) http://cruisecontrolrb.thoughtworks.com/

Add cruise task to Rakefile

```
# cd to Rails project dir
$ cd ~/mysite
$ gedit Rakefile
# Add cruise task to bottom after 'requires':
task :cruise do
  Rake::Task['test'].invoke
end
$ svn commit Rakefile -m "add cruise task"
# Check cruise webpage, should still be passing
```

Tweak firefox for automation
open firefox, navigate to 'about:config'
search for 'browser.sessionstore.resume_from_crash'
toggle to false
Preferences - Tabs - uncheck ''warn when closing multiple tabs'
Maybe turn off update prompts too...

E. JsUnit Setup

Download and Unzip JsUnit

```
$ cd
$ wget
http://easynews.dl.sourceforge.net/sourceforge/jsunit
/jsunit2.2alpha11.zip
$ unzip jsunit2.2alpha11.zip
# copy junit.jar file to Ant lib dir (required by Ant)
$ sudo cp jsunit/java/lib/junit.jar /usr/share/ant/lib/
```

Copy jsunit to your app and check in

```
$ cd mysite/public/javascripts
$ mv /home/ci/jsunit .
$ svn add jsunit
$ export EDITOR=gedit
$ svn propedit svn:ignore jsunit/logs
# add * to ignore list
$ svn propedit svn:executable jsunit/bin/unix/start-firefox.sh
# enter “true”
$ svn commit -m "add jsunit"
```

Create a jsunit test

```
$ mkdir test_pages
$ gedit test_pages/prototype_test.html
<html>
<head>
  <script language="JavaScript"
type="text/javascript"
src="../jsunit/app/jsUnitCore.js"></script>
  <script language="JavaScript"
type="text/javascript" src="../prototype.js"></script>
  <script language="javascript">
    function testPrototypeWordSplit() {
      string = 'one two three';
      assertEquals('one', ($w(string))[0]);
    }
  </script>
</head>
<body></body>
</html>
```

Run the jsunit test manually from browser and commit

```
$ cd
$ cd mysite
$ ruby script/server # unless you still have it running

$ firefox http://localhost:3000/javascripts/jsunit/testRunner.html
# Enter this in the "Run" field and click “Run”: http://localhost:3000/javascripts/test_pages/prototype_test.html
$ svn add public/javascripts/test_pages
$ svn commit -m "jsunit test"
```

Take this opportunity to familiarize yourself with JsUnit and JsUnit Server. It's not covered here ;) http://jsunit.net/

"Punt" and make a manual jsunit_start_server script
# Because automated process management is not TSTTCPW for this tutorial, and it's hard
# This is also easily ported to a batch file on windows
$ cd mysite
$ gedit script/jsunit_start_server.sh
ant -f /home/ci/mysite/public/javascripts/jsunit/build.xml -DbrowserFileNames= /home/ci/mysite/public/javascripts/jsunit/bin/unix/start-firefox.sh -Dport=8081 start_server

Check in jsunit_start_server script and leave it running

```
$ svn add script/jsunit_start_server.sh
$ svn propedit svn:executable
script/jsunit_start_server.sh
# add 'true' line
$ script/jsunit_start_server.sh
# ignore warning about tools.jar
# make sure it starts and leave it running
# ctrl-c if you want to kill it
# open a new terminal tab
$ svn ci -m "add jsunit start script"
```

Add jsunit task

```
$ gedit Rakefile
task :cruise do
  Rake::Task['test'].invoke
  Rake::Task['jsunit_distributed_test'].invoke
end

task :jsunit_distributed_test do
  output = `ant -f public/javascripts/jsunit/build.xml
-Durl=http
://localhost:8080/jsunit/jsunit/testRunner.html?testPa
ge=/jsunit/test_pages/prototype_test.html
-DremoteMachineURLs=http://localhost:8081
-DresourceBase=public/javascripts distributed_test`
  raise "JsUnit Failed:\n" + output unless
$?.success?
  puts "JsUnit tests passed"
end
```

Commit jsunit task and check cruise
Open cruise webpage under mozilla
jsunit will kill firefox, so we need a different browser
$ /usr/local/mozilla/mozilla http://localhost:3333
# if you want, add a quick launch for mozilla: right click -> add to panel -> custom application launcher
$ svn commit Rakefile -m "add jsunit_distributed_test task"
# Check cruise webpage, should still be passing

F. Selenium Setup

Selenium 0.8.1 is proven, 0.9.0 has had problems. Latest unreleased version is reported to be OK.

Download Selenium Remote Control

```
$ cd
$ wget http://release.openqa.org/selenium-remote-control/0.8.1/selenium-remote-control-0.8.1.zip
$ unzip selenium-remote-control-0.8.1.zip
```

Make a manual selenium_start_server script

```
$ cd mysite
$ cp /home/ci/selenium-remote-control-0.8.1/server/selenium-server.jar lib
$ svn add lib/selenium-server.jar
$ gedit script/selenium_start_server.sh
java -jar /home/ci/mysite/lib/selenium-server.jar -interactive
$ svn add script/selenium_start_server.sh
$ export EDITOR=gedit
$ svn propedit svn:executable script/selenium_start_server.sh
# add 'true' line
$ script/selenium_start_server.sh
# make sure it starts and leave it running, ctrl-c to kill it
# Open new terminal tab
$ svn ci -m "add selenium start script and jar"
```

Set up selenium test dir and copy ruby API file

```
$ cd mysite
$ mkdir test/selenium
$ cp ~/selenium-remote-control-0.8.1/ruby/selenium.rb test/selenium
```

Create selenium test stub

```
$ gedit test/selenium/user_test.rb
require 'test/unit'
require File.expand_path(File.dirname(__FILE__) + '/selenium')

class UserTest < Test::Unit::TestCase
  def setup
    @selenium =
Selenium::SeleneseInterpreter.new("localhost", 4444, "*firefox
/usr/lib/firefox/firefox-bin", "http://localhost:3001/", 10000);
    @selenium.start
  end

  def teardown
    @selenium.stop
  end

  def test_user_add_flow
  end
end
```

Fill in selenium test stub

```
$ gedit test/selenium/user_test.rb
  def test_user_add_flow
    timestamp = Time.new.to_s
    user_name = 'joe ' + timestamp
    @selenium.open "http://localhost:3001/users"
    @selenium.click "link=New user"
    sleep 2 # <- Sleeping is bad! Use a wait_for loop...
    @selenium.type "id=user_name", user_name
    @selenium.click "commit"
    sleep 2
    assert @selenium.is_text_present(user_name)
  end
```

Create selenium_test rake task including start and stop of server

```
$ gedit Rakefile
task :cruise do
  ...
  Rake::Task['selenium_test'].invoke
end

task :selenium_test do
  begin
    process = IO.popen("ruby
/home/ci/cc/projects/MySite/work/script/server --port=3001")
    output = `ruby test/selenium/user_test.rb`
    raise "Selenium Failed:\n" + output unless $?.success?
    puts "Selenium tests passed"
  ensure
    Process.kill(9,process.pid)
  end
end
```

Check in and check cruise

```
$ svn add test/selenium
$ svn commit -m "selenium test"
# check cruise, it should run everything and be green
```

Break tests and fix them!
cause ruby/jsunit/selenium failures, and check them in
see cruise go red, then fix them
click links for ruby/selenium failures
there's a test bug! (next page after too many tests)
good to drop DB before each CI run...
This naive implementation has return code bugs (crash if webrick already running)

Same concept for other tools/ Languages/ CI Engines

Coding Done!

2. Gettin' Fancier

All Handwaving Now

Multiplatform

Multibrowser

Farms

Virtualization: One Box, Three Platforms mac/win/linux

Automate and Test Deployment Process

Test Rollback process!

Configuration Management / Version Control

Auto-tag Green Builds

Automatically pre-create Release Branches

Build <u>ALL</u> active branches under CI

Multiple Libraries/ Projects

Dependencies Among Common Libraries and Projects

Dependency modifications should trigger builds of all dependents

Consistent Tags/Baselines Among Projects: Naming/Usage

Versioning of Dependencies (or not):

Mainline / Snapshot / trunk / HEAD
vs
baselines / tags

Different Builds for Different Environments: Development vs Demo/Prod

Publishing Artifacts/ Dependencies:

Deployed (Jars/Gems) vs SCM (svn:externals)

Hackability vs Stability: Fear should not inhibit improvement of common libraries

Optimism vs Pessimism: Do What dependency versions are you deploying to prod?

Nirvana: Green tags/artifacts instantly used across all dev environments, all deploys have known, green, stable, baselined dependencies

Suites: You can have more than one!

It's all about Feedback

Timely
vs
Comprehensive

Fast
vs
Thorough

Commit-Triggered vs Scheduled

Minimize Checkout Time

Get HUGE Dependencies and binaries out of Source Control

RubyGems / Maven vs svn:externals / CVS modules

Metrics

Code Coverage - Emma/rcov

Mutation Testing – Heckle, Jester

red/green trends

Build Length Trends

Notification

Information Radiator(s)

email

RSS

IM

Growl

Ambient Orb

13” CRT with red/green background

Whatever people will pay attention to!

3. Gotchas

Random Gotchas / Mantras:

* “It's not easy being Green”
* Broken Windows are Bad (“Who cares, it's always red...”)
* False Negatives are Bad
* Crying Wolf (“it failed for no reason”)
* “Intermittent” failures (but it's not intermittent after you can reproduce it)
* “Works Locally” (is your local environment the same as CI? Which one is Prod closer to???)
* You can always “temporarily” disable a test in CI
* One disabled test is better than a red CI
* False Positives are Bad too - being Green, when return code (echo $?) from some step is not 0
* Browser Settings (autoupdate, etc) Preventing Browser Close

4. Questions?

Chad Woolley

thewoolleyman @ gmail.com

thewoolleyweb.com/ci_for_the_web_2.0_guy_or_gal