# **Introduction**

This tutorial will explain how to get started with Juju, including installing, configuring, and bootstrapping a new Juju environment. Prerequisites include:

* An Ubuntu, OSX, or Windows machine to install the client upon.
* An environment which can provide a new server with an Ubuntu cloud operating system image on-demand. See under Install & Configure on the left for how to use different providers, including any provider-specific settings.
* An SSH key-pair. On Linux and Mac OSX: ssh-keygen -t rsa -b 2048 On Windows: See the [Windows instructions for SSH and PuTTY](https://jujucharms.com/docs/stable/getting-started-keygen-win).

# **Installation**

## **Ubuntu**

To install Juju, you simply need to grab the latest juju-core package from the PPA:

sudo add-apt-repository ppa:juju/stable  
sudo apt-get update && sudo apt-get install juju-core

For more information on installing and the current versions available, see [the releases page](https://jujucharms.com/docs/stable/reference-releases).

## **Mac OSX**

Juju is in [Homebrew](http://brew.sh/), to install do:

brew install juju

We also recommend trying Juju in [our Vagrant box](https://jujucharms.com/docs/stable/config-vagrant).

For more installation information and what versions are available, see [the releases page](https://jujucharms.com/docs/stable/reference-releases).

## **Windows**

See [the releases page](https://jujucharms.com/docs/stable/reference-releases) to download and run the latest version of the Juju Windows installer.

We also recommend trying Juju in [our Vagrant box](https://jujucharms.com/docs/stable/config-vagrant).

# **Configuring**

Juju needs to be configured to use your cloud provider. This is done via the following file:

## **Linux & Mac OSX**

~/.juju/environments.yaml

## **Windows**

%APPDATA%\Juju\environments.yaml

Where %APPDATA% is typically defined as C:\Users\<user>\AppData\Roaming.

Juju can automatically generate the file in this way:

juju generate-config

This action will not overwrite an existing file but merely dump the information onscreen (STDOUT). It will contain sample profiles for different types of cloud services. Edit it to provide specific information for your chosen cloud provider. For more specifics on what needs to be changed, see the relevant sections in the left pane (under *Install & Configure*).

**Note:** Juju's command line interface includes documentation, running juju help will show you the topics. You can also look at the [Juju command cheatsheet](https://github.com/juju/cheatsheet) if you are looking for a convenient command guide.

# **Testing your setup**

The first step is to create a bootstrap environment. This is a cloud instance that Juju will use to deploy and manage services. It will be created according to the configuration you have provided, and your public SSH key will be uploaded automatically so that Juju can communicate securely with the bootstrap instance.

juju bootstrap

**Note:** If you have multiple environments configured, you can choose which one to address with a particular command by adding the -e switch followed by the environment name, E.g. -e hpcloud.

You may have to wait a few moments for this command to return, as it needs to perform various tasks and contact your cloud provider.

Assuming it returns successfully, we can now deploy some services and explore the basic operations of Juju.

To start with, we will deploy WordPress:

juju deploy wordpress

Juju will download and use the WordPress charm, through the bootstrap instance, to request and deploy whatever resources it needs to install this service.

Since WordPress requires a database, we will deploy one:

juju deploy mysql

Again, Juju will do whatever is necessary to deploy this service for you, and it may take some time for the command to return.

**Note:** If you want to get more information on what is actually happening, or to help resolve problems, you can add the --show-log switch to the juju command to get verbose output.

Although we have deployed WordPress and a MySQL database, they are not linked together in any way yet. To do this we run:

juju add-relation wordpress mysql

This command uses information provided by the relevant charms to associate these services with each other in whatever way makes sense. There is much more to be said about linking services together which is covered in the Juju command documentation, but for the moment, we just need to know that it will link these services together.

In order to make our WordPress public, we now need to expose this service:

juju expose wordpress

This service will now be configured to respond to web requests, so visitors can see it. But where exactly is it? If we run the juju status command, we will be able to see what services are running, and where they are located.

juju status

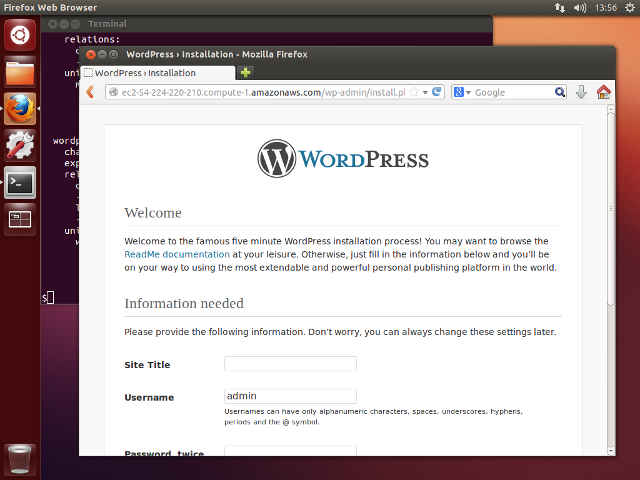
The output from this command should look something like this:

machines:  
 "0":  
 agent-state: started  
 agent-version: 1.10.0  
 dns-name: ec2-50-16-167-135.compute-1.amazonaws.com  
 instance-id: i-781bf614  
 series: precise  
 "1":  
 agent-state: started  
 agent-version: 1.10.0  
 dns-name: ec2-23-22-225-54.compute-1.amazonaws.com  
 instance-id: i-9e8927f6  
 series: precise  
 "2":  
 agent-state: started  
 agent-version: 1.10.0  
 dns-name: ec2-54-224-220-210.compute-1.amazonaws.com  
 instance-id: i-5c440436  
 series: precise  
services:  
 mysql:  
 charm: cs:precise/mysql-18  
 exposed: false  
 relations:  
 db:  
 - wordpress  
 units:  
 mysql/0:  
 agent-state: started  
 agent-version: 1.10.0  
 machine: "1"  
 public-address: ec2-23-22-225-54.compute-1.amazonaws.com  
 wordpress:  
 charm: cs:precise/wordpress-12  
 exposed: true  
 relations:  
 db:  
 - mysql  
 loadbalancer:  
 - wordpress  
 units:  
 wordpress/0:  
 agent-state: started  
 agent-version: 1.10.0  
 machine: "2"  
 public-address: ec2-54-224-220-210.compute-1.amazonaws.com

There is quite a lot of information here. The first section, titled **machines:**, details all the instances which are currently running. For each you will see the version of Juju they are running, their hostname, instance id and the series or version of Ubuntu they are running.

After that, the sections list the services which are currently deployed. The information here differs slightly according to the service and how it is configured. It will, however, always list the charm that was used to deploy the service, whether it is exposed or not, its address and what relations exist.

From this readout, we can see that WordPress is exposed and ready. If we point a web browser at the address we should be able to access it:



Congratulations, you have just deployed a service with Juju!

Now you are ready to deploy whatever service you want from the 100s available at the [Juju Charm Store.](https://jujucharms.com/).

To remove all current deployments and clear everything in your cloud, you can run:

juju destroy-environment <environment-name>

Where <environment-name> is the name you gave the environment when you configured it. A warning will be displayed and the user will be prompted whether or not to continue. This action will remove everything in the specified environment, including the bootstrap node.

See the [charm documentation](https://jujucharms.com/docs/stable/charms) to learn more about charms, including configuring options and managing running systems.