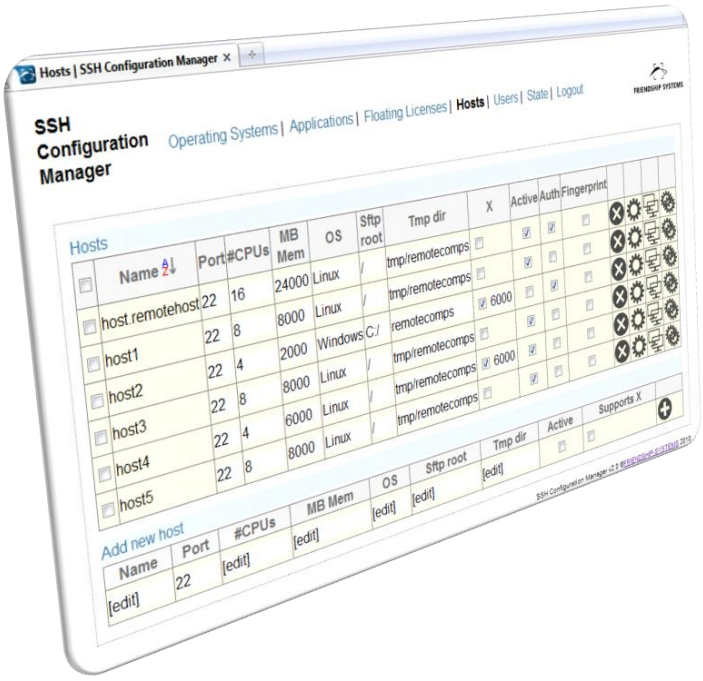
FRIENDSHIP SYSTEMS
A GL company

Sample Setup Walkthrough

FRIENDSHIP-Framework SshConfigurationManager

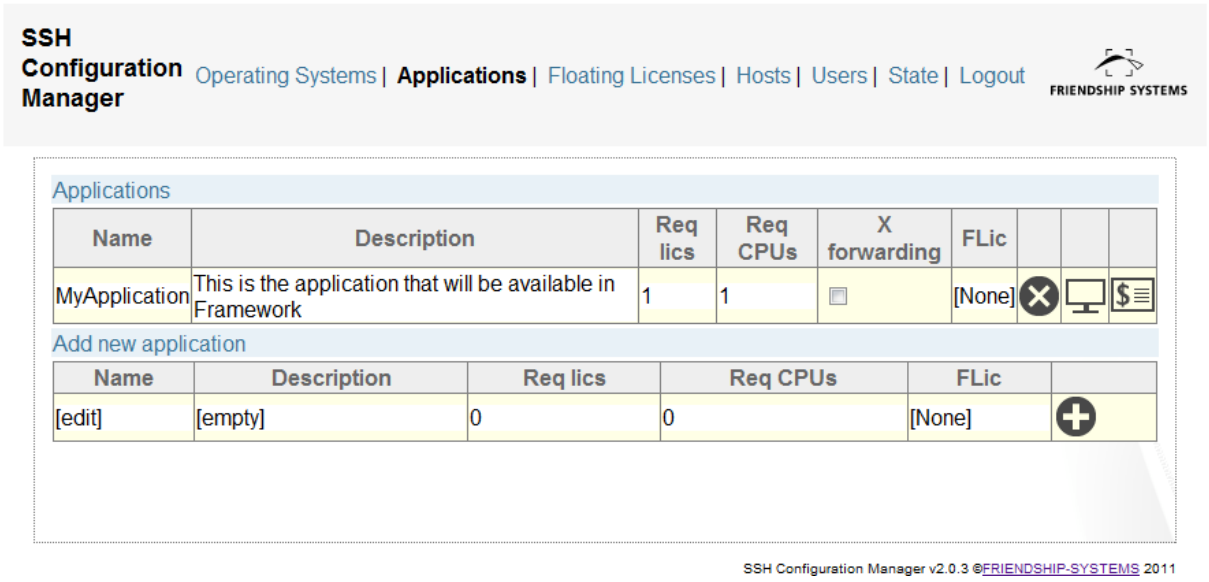


SshConfigurationManager

The SshConfigurationManager is the configuration interface for the SshResourceManager, a grid engine that was specifically designed to allow the FRIENDSHIP-Framework to execute external programs on remote computers. This document provides a quickstart walkthrough for the required settings in the SshConfigurationManager. It is an addition to the adminguide which provides additional information especially regarding the installation and setup process for both, the SshResourceManager and the SshConfigurationManager. Please make sure to study the adminguide carefully. This

1. Setting up an application

The application is the interface that is provided to the user of the FRIENDSHIP-Framework. When the SshResourceManager is used, the FRIENDSHIP-Framework user can choose from the installed applications to determine which program to execute. This sample setup has one application configured which is called “MyApplication”.

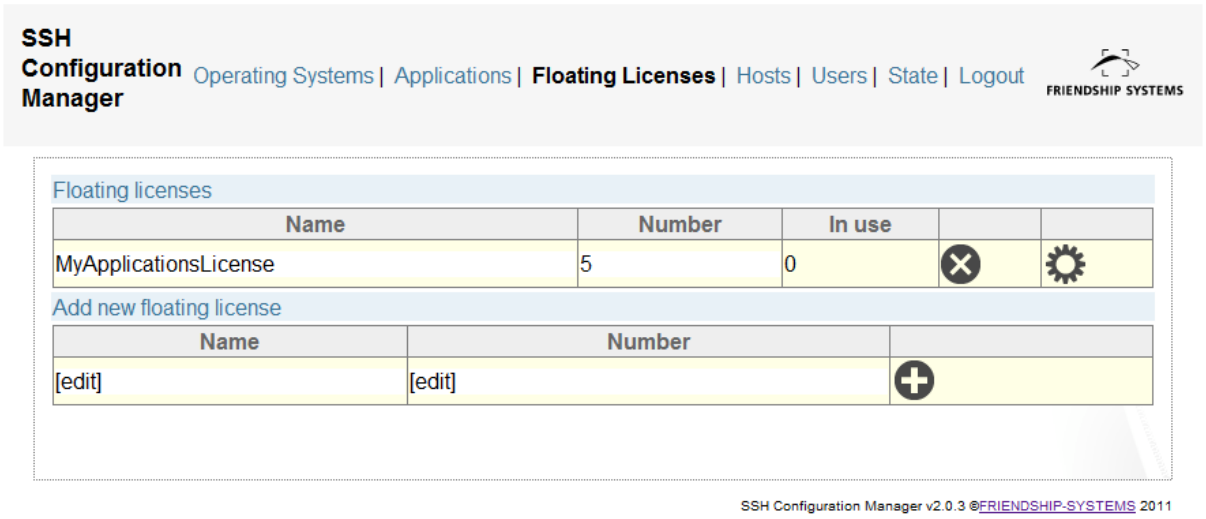


An application requires the following inputs:

- Name – This allows users to identify which program to execute. The name should support the user in choosing the right application.
- Description – Additional information about an application can be provided here. If in doubt, the user may find it useful to choose the correct application if the name is not descriptive enough.
- Req lics – This field holds the number of licenses that are needed to execute the program once. For most programs this should be set to 1, however, some applications may acquire more than one license per instance.
- Req CPUs – Similar to the Req lics setting, this holds the number of processors one instance of an application uses when it runs. Again, a setting of 1 is probably the most common case; however, some multithreaded applications may need more.
- X forwarding – Some applications may need an X server to function properly. If that is the case, this setting needs to be applied. Please note that such an application should only be installed on a host that provides X forwarding (see the section Hosts below) and the X forwarding settings of the SshResourceManager need to be properly configured (see the adminguide for details).
- FLic – The SshResourceManager knows two types of licenses: Hard licenses (aka Node Locked Licenses) and Floating licenses. While Hard licenses are configured for a particular host (see section Hosts below), Floating licenses are shared among all hosts that have an application installed. When a floating license is configured (see section Floating Licenses below) it can be chosen here.

2. Setting up a Floating License

The SshResourceManager allows to configure license information for the applications and hosts it manages. The number of available licenses determines how many instances of a particular application can run at the same time. In this sample setup we will set up a floating license for the application we configured in step 1.



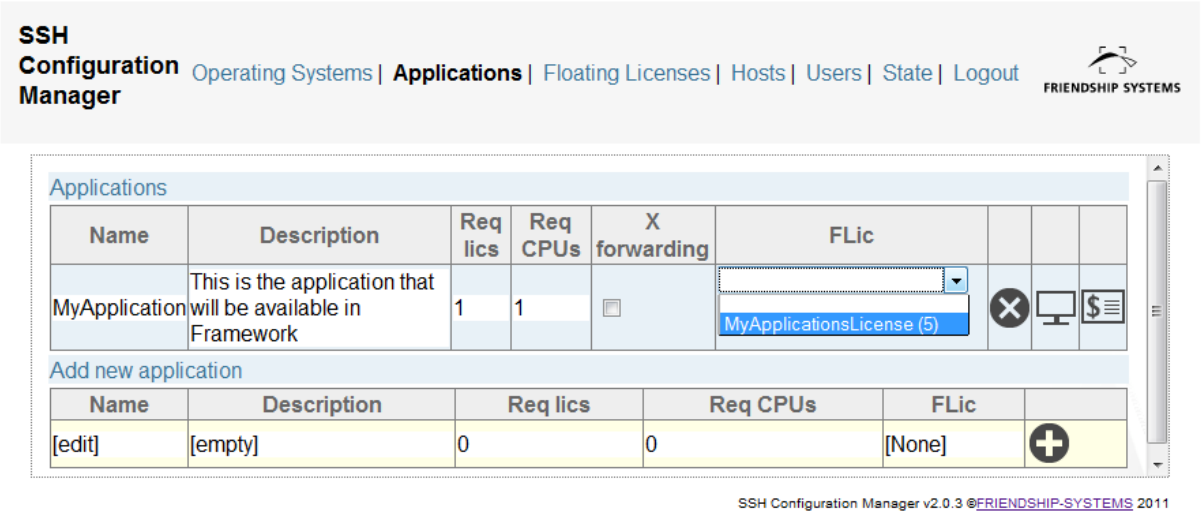
A floating license only has two possible settings:

- Name – A name that is used for identification
- Number – The number of available slots of this floating license

Please note that the SshResourceManager can only keep track of the licenses it uses. It does not connect to a Floating License Server (e.g. FlexLM) to check whether free floating license slots are available.

3. Adding the Floating License to the Application

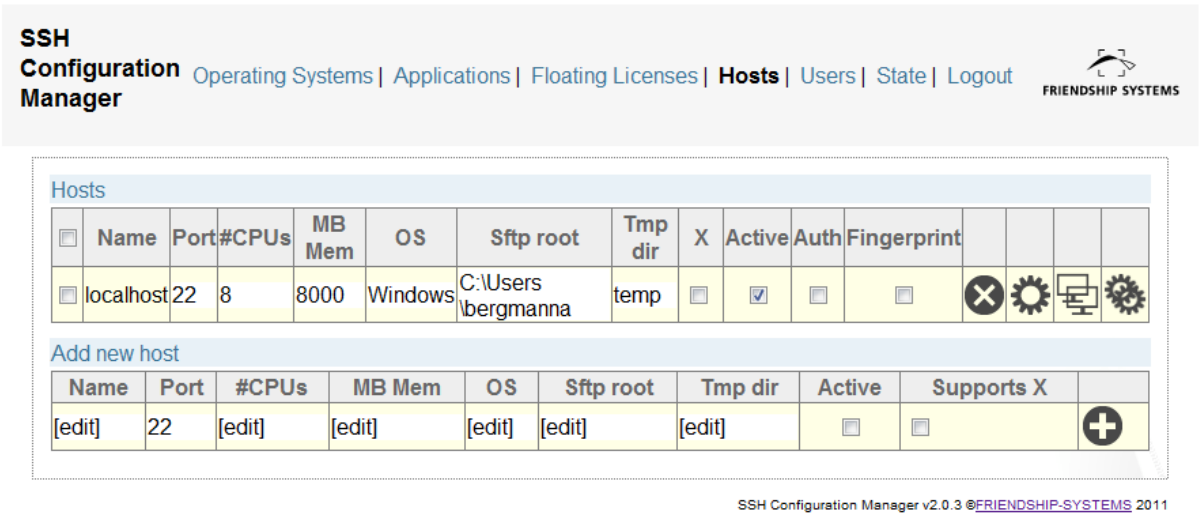
After configuring the floating license it can be associated with the application. We go back to the Applications page and do so:



We only need to select the desired floating license from the list. For convenience, you can also create a floating license directly on the applications page by clicking on the last icon in the application's row.

4. Adding a host

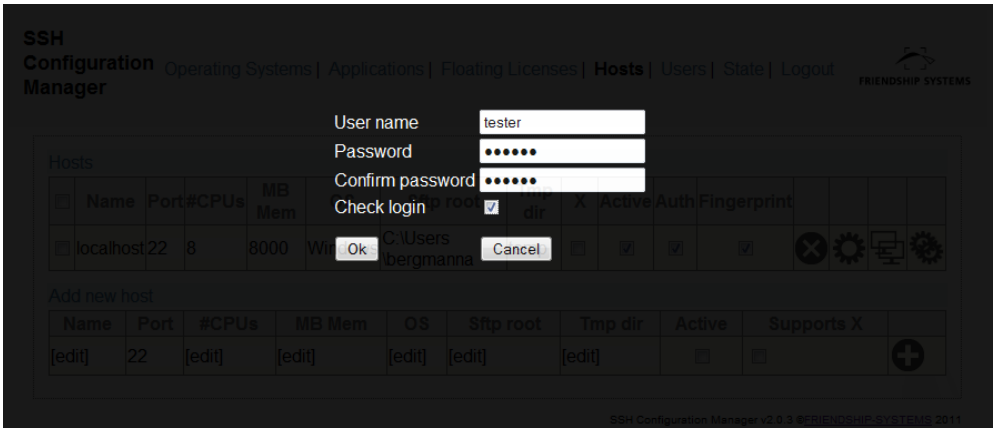
Hosts are the computers that execute the external program. They are connected to via SSH from the SshResourceManager, so they need to have an SSH server running. The Hosts are your Computational Grid. In this sample setup we configure one host, the localhost.



There are several settings that need to be applied here:

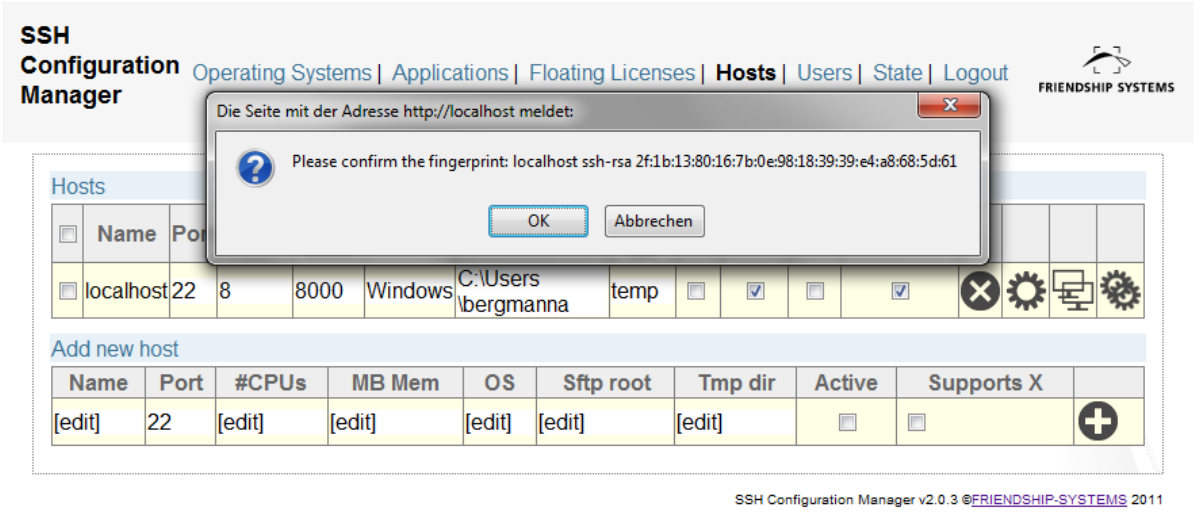
- Name – The name is the host's name in the network. An IP address can also be used, however, that should only be done if the IP address does not change through time.
- Port – This is the port where the SSH server is running (default 22).
- #CPUs – This is the number of processors or cores that are installed in this host. This is related to the "Req CPUs" setting of the application. The SshResourceManager keeps track of the number of CPUs used on a particular host and matches the number of available CPUs with the application's setting in order to determine whether that host can currently execute the application.
- MB Mem – This is the amount of memory the host has in MB.
- OS – Some operations the SshResourceManager needs to perform on the host are dependent on the operating system of the host, so this setting holds the operating system the host is running. By default Windows and Linux are configured as available operating systems, more can be added on the Operating Systems page.
- SFTP root – This is the directory an SFTP connection to this host is initially routed to. The correct value for this property depends on the setting of the SSH/SFTP server. For Linux hosts this is usually the user's HOME directory. If that is the case, the recommended setting for SFTP root is "." (without quotes, just the dot). On Windows hosts it highly depends on the SSH/SFTP server.
- Tmp dir is a directory relative to the SFTP root directory which will be created for temporary data. It is automatically created if it does not exist.
- X – If this host supports X forwarding, this should be checked. If this is checked a comma separated lists of ports the X server is forwarded to can be supplied.
- Active – Allows to activate and deactivate hosts.

The Auth setting allows to provide SSH login credentials for this host. As an alternative those credentials can also be set in the ResourceManagerSetupSsh object inside the FRIENDSHIP-Framework. This is a convenience setting so users do not have to worry about the SSH login. After clicking the checkbox, the following screen will come up:



Additionally to the username and password the SshResourceManager can be asked to validate the credentials in order to make sure that the login succeeds.

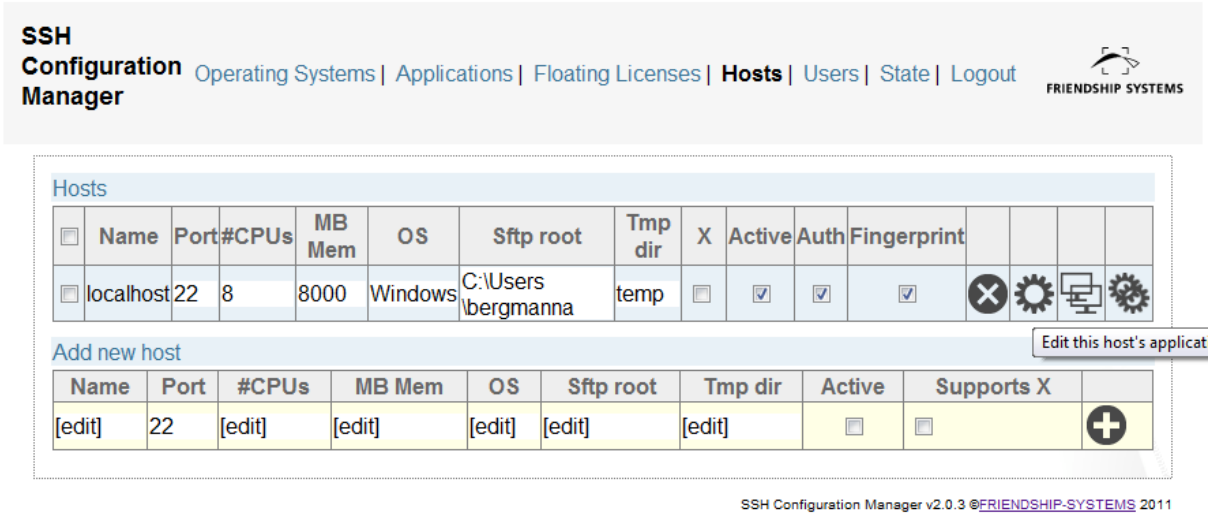
For security reasons the SshResourceManager checks the identity of a Host whenever it connects to it. This is to avoid man-in-the-middle attacks and IP spoofing. Therefore, we will need to confirm the host's fingerprint once by clicking the "Fingerprint" checkbox:



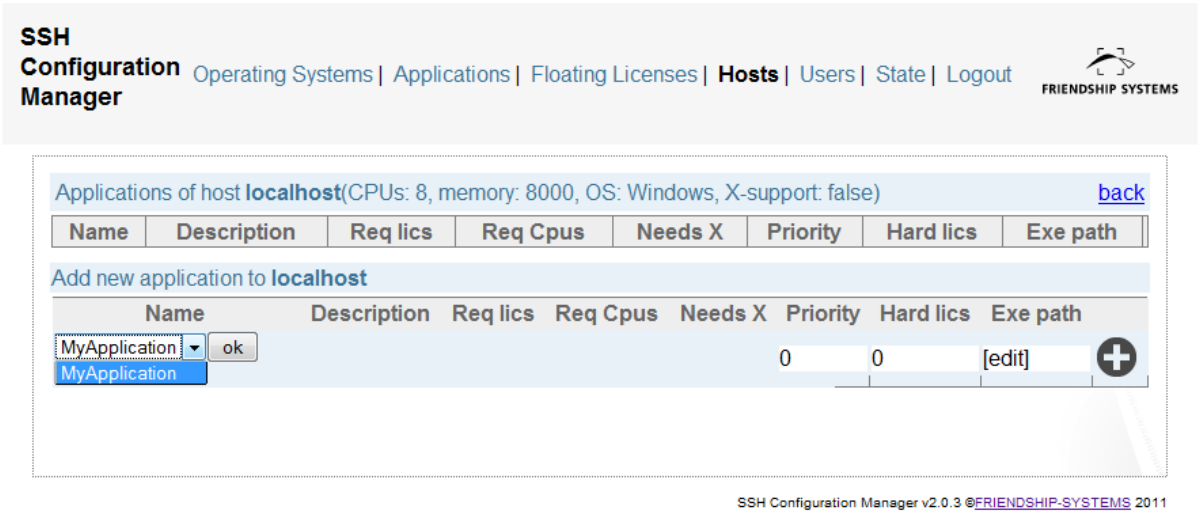
After checking the fingerprint it can be confirmed.

5. Adding the application to the host

Now we can add the application to the host. We click on the single gear icon in the host's row to get to the appropriate view:



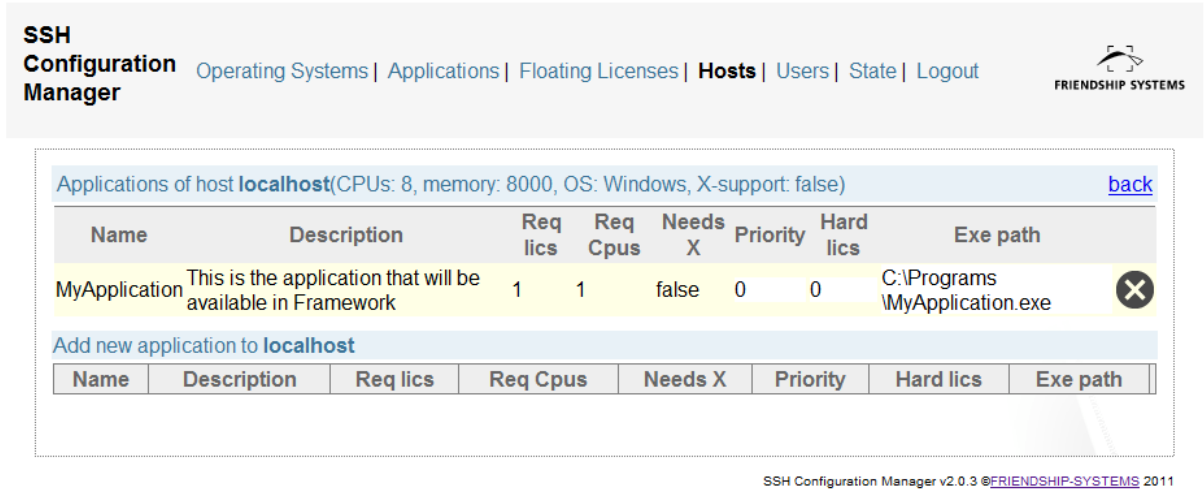
Now we select the previously configured application "MyApplication" from the list of available applications.



The most important setting is the path to the executable ("Exe path"). This is the command that will be executed on the host when the SshResourceManager decides to use this host to execute the program.

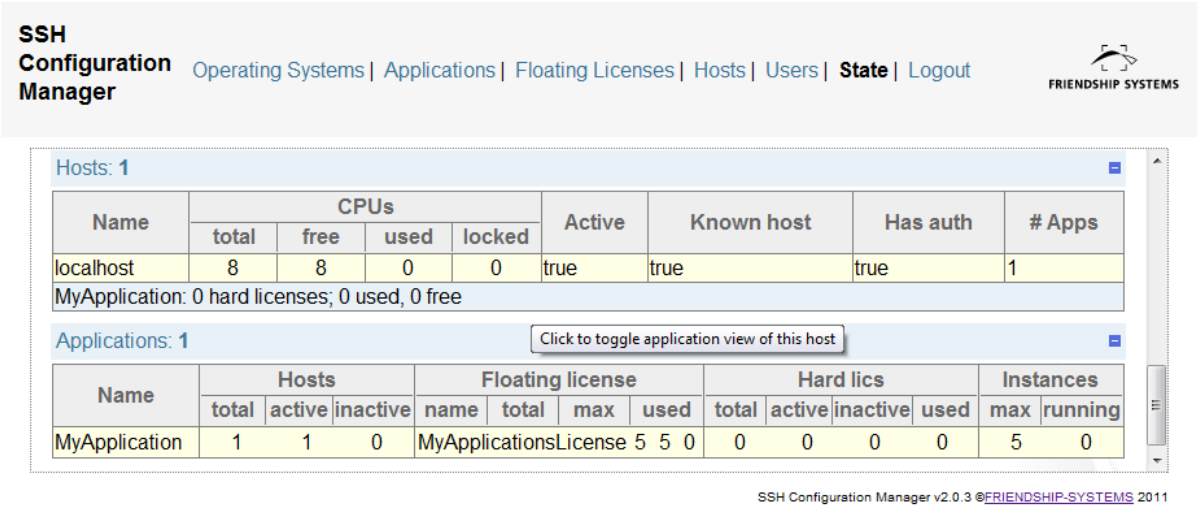
There are two other possible settings:

- Priorities – Depending on the configuration of the SshResourceManager (see adminguide, section Scheduler Settings) this allows to prioritize this host when the chosen application is to be started.
- Hard lics – This is the number of Hard licenses (aka Node Locked Licenses) that are available for this application on this host. Since we have configured a floating license for our application, this is not used in this sample setup. Note that it is possible to use both, hard licenses and floating licenses, combined.



6. Check the setup

The setup is done. We can now go to the state page, to check the setup:



SSH Configuration Manager Operating Systems | Applications | Floating Licenses | Hosts | Users | **State** | Logout

FRIENDSHIP SYSTEMS

Hosts: **1**

Name	CPUs				Active	Known host	Has auth	# Apps
	total	free	used	locked				
localhost	8	8	0	0	true	true	true	1
MyApplication: 0 hard licenses; 0 used, 0 free								

Applications: **1**

Click to toggle application view of this host

Name	Hosts			Floating license				Hard lics				Instances	
	total	active	inactive	name	total	max	used	total	active	inactive	used	max	running
MyApplication	1	1	0	MyApplicationsLicense	5	5	0	0	0	0	0	5	0

SSH Configuration Manager v2.0.3 ©FRIENDSHIP-SYSTEMS 2011

Here we see that currently 5 instances of the application “MyApplication” could be executed in parallel. The limiting factor is the number of available licenses. If there would be more that 8 available licenses, the number of instances would be limited by the number of available processors of the hosts, in our case 8.