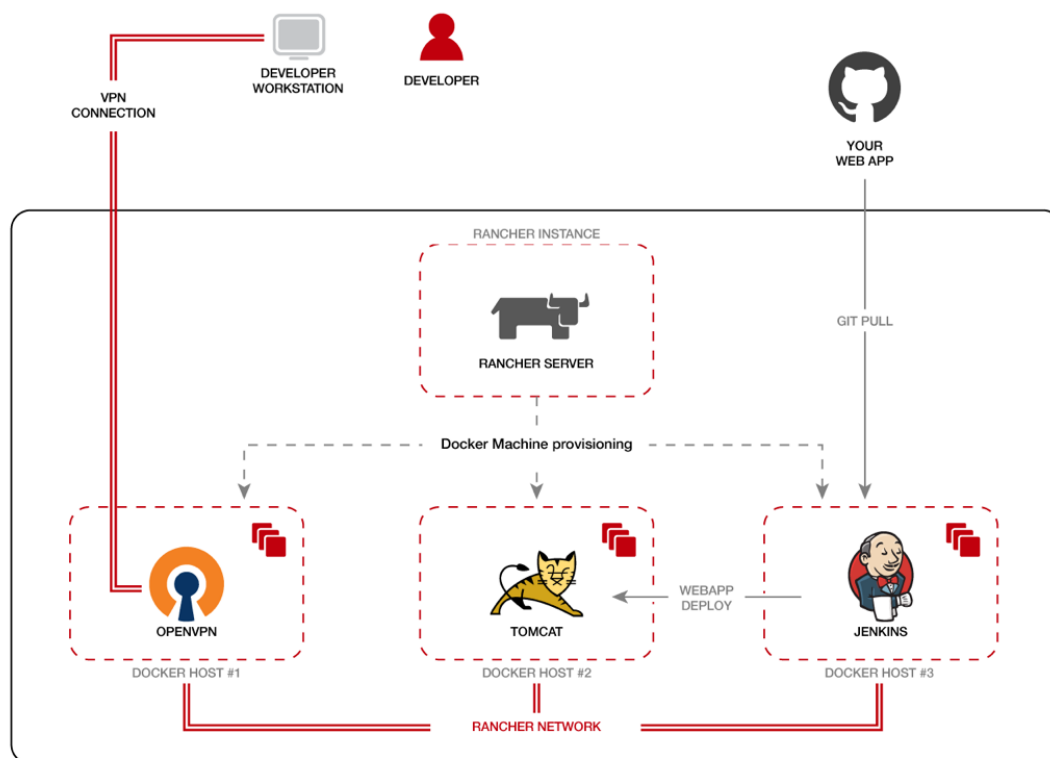


# Creating a safe dev environment on a public Cloud with Docker, Jenkins, OpenVPN and Rancher

Marc-Aurèle Brothier

This setup will use 4 instances to run Rancher server, Jenkins, Tomcat and an OpenVPN server. Then you will be able to work, build, deploy over a VPN link to instances in the cloud.



## Setup our cloud

### Get our credentials

Log on our [cloud provider](#) portal and get the [account API credentials](#) which you need to expose as environment variables, in my case using exoscale:

```
export DEMO_API_KEY='<your api key>'
export DEMO_SECRET_KEY='<your secret key>'
export CLOUDSTACK_ENDPOINT='https://api.exoscale.ch/compute'
export CLOUDSTACK_KEY=$DEMO_API_KEY
export CLOUDSTACK_SECRET=$DEMO_SECRET_KEY
```

### Create a security group

Create a security group by using this [python script](#):

```
# Create a virtual environment named ".venv"
python3 -m venv .venv
source ./venv/bin/activate
# Install the cs package ①
pip install cs
# Deploy firewall rules
./scripts/create-security-group.py "demo-devovx"
```

① <https://pypi.python.org/pypi/cs>

Or manually create one in the interface with those rules:

- Allow 22/tcp, 2376/tcp and 8080/tcp ports from any source, needed for Docker machine to provision hosts.
- Allow 500/udp and 4500/udp ports from any source, needed for Rancher network.
- Allow 9345/tcp and 9346/tcp ports from any source, needed for UI features like graphs, view logs, and execute shell.
- Allow 1194/tcp and 2222/tcp ports from any source, needed to publish our VPN server container.
- Allow 443/tcp ports from any source, needed to access Rancher UI over HTTPS.

security groups / detail

credit: 50.00 CHF Demo PeakXL ?

### Rules of demo-devovx

PING SSH RDP NEW RULE

Type	Protocol	Source	Port	DELETE SELECTED
<input type="checkbox"/> INGRESS	TCP	CIDR: 0.0.0.0/0	TCP 22 - 22	<input type="checkbox"/>
<input type="checkbox"/> INGRESS	TCP	CIDR: 0.0.0.0/0	TCP 2376 - 2376	<input type="checkbox"/>
<input type="checkbox"/> INGRESS	TCP	CIDR: 0.0.0.0/0	TCP 8080 - 8080	<input type="checkbox"/>
<input type="checkbox"/> INGRESS	UDP	CIDR: 0.0.0.0/0	UDP 500 - 500	<input type="checkbox"/>
<input type="checkbox"/> INGRESS	UDP	CIDR: 0.0.0.0/0	UDP 4500 - 4500	<input type="checkbox"/>
<input type="checkbox"/> INGRESS	TCP	CIDR: 0.0.0.0/0	TCP 9345 - 9345	<input type="checkbox"/>
<input type="checkbox"/> INGRESS	TCP	CIDR: 0.0.0.0/0	TCP 9346 - 9346	<input type="checkbox"/>
<input type="checkbox"/> INGRESS	TCP	CIDR: 0.0.0.0/0	TCP 1194 - 1194	<input type="checkbox"/>
<input type="checkbox"/> INGRESS	TCP	CIDR: 0.0.0.0/0	TCP 2222 - 2222	<input type="checkbox"/>

Figure 1. Security Group

# Create a docker machine

1. Start a new docker machine directly from your local machine by using a docker driver since exoscale is providing one:

```
docker-machine create --driver exoscale \
  --exoscale-api-key $DEMO_API_KEY \
  --exoscale-api-secret-key $DEMO_SECRET_KEY \
  --exoscale-instance-profile 'Tiny' \
  --exoscale-disk-size '50' \
  --exoscale-image 'ubuntu-16.04' \
  --exoscale-security-group 'demo-devovx' \
  --exoscale-availability-zone 'ch-dk-2' \
  "devovxuk-ma"
```

2. Export the docker machine environment variables for this new machine:

```
eval $(docker-machine env devovxuk-ma)
```

3. Start a docker container with Rancher server on our new docker machine:

```
docker run -d -p 8080:8080 rancher/server:v1.1.0-dev3
```

4. Point your browser to the public IP found in [the portal](#) for our instance: <http://xxx.xxx.xxx.xxx:8080/> (<http://159.100.249.155:8080/> in my case).

## TLS

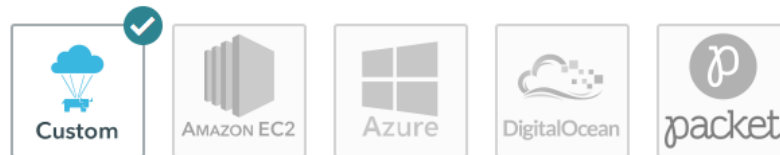
### NOTE

At the time of writing, there's a bug in the rancher server UI which loads content over http only, making impossible to start a NGINX-TLS proxy in front of it. You'll find in [docker-compose.yml](#) file the setup to run rancher behind a nginx + let's encrypt proxy.

## Provisioning Docker hosts

1. First you need to add our cloud provider, exoscale in my case, as it's not enabled by default.

## Hosts: Add Host



Manage available machine drivers























Figure 2. Only a few provider are enabled by default

2. Click on *Manage available machine drivers* and click the *play* sign button to the far right to enable exoscale driver.

### Machine Drivers

Additional `docker-machine` drivers can be loaded here and used in the Add Host screen.

 Add Machine Driver

State	Name	Driver URL	UI URL	Checksum	
Active	amazonec2	Built-In	Built-In	Built-In	 
Active	azure	Built-In	Built-In	Built-In	 
Active	digitalocean	Built-In	Built-In	Built-In	 
Active	exoscale	Built-In	Built-In	Built-In	 
Inactive	generic	Built-In	Other	Built-In	 
Inactive	google	Built-In	Other	Built-In	 
Inactive	hyperv	Built-In	Other	Built-In	 
Inactive	openstack	Built-In	Other	Built-In	 
Active	packet	.../v0.1.2/docker-machine-driver-pack...	Built-In	cd610cd7d9...	 
Inactive	rackspace	Built-In	Built-In	Built-In	 
Inactive	softlayer	Built-In	Other	Built-In	 
Inactive	ubiquity	.../v0.0.2/docker-machine-driver-ubiq...	Built-In	7fba983dfdb...	 
Inactive	vmwarevcloudair	Built-In	Other	Built-In	 
Inactive	vmwarevsphere	Built-In	Built-In	Built-In	 

3. Go back in your browser and you will see the logo for exoscale, click on it and enter your credentials, the same as used before for docker.

## Hosts: Add Host



Manage available machine drivers

### ACCOUNT ACCESS

API Key

Your Exoscale API Key

Secret Key

Your Exoscale secret key

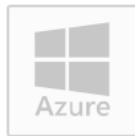
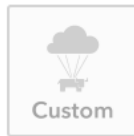
Paste in your Exoscale key pair here. We'll use this key to create your new Instances.

Next: Authenticate & select a Security Group

Cancel

- Next choose the security group you created in the portal (*might be buggy*)

## Hosts: Add Host



Manage available machine drivers

### ACCOUNT ACCESS

API Key EXOf2d35248bf3acf1302e6ed7

Secret Key *Provided*

### SECURITY GROUP

Security Group ☐ Standard: Automatically create a **rancher-machine** group  
☒ Custom: Choose an existing group

demo-devxxx

For Rancher to work correctly your security group will need to allow traffic:

- From the Rancher server to **TCP** port **22** (SSH to install and configure Docker)
- From and To all other hosts on **UDP** ports **500** and **4500** (for IPsec networking)
- These rules will **not** be added automatically.

Next: Set Instance options

Cancel

5. Next choose the number of instances, 3 for our scenario (VPN, Jenkins, Tomcat). Named them as you want and choose a profile size that suits your needs.

INSTANCE

Quantity

3

Name

devoux-exo-

Description

Description

Instance Profile

Tiny 1024mb 1cpu

Root Size

50

Hosts will be named devoux-exo-1 – devoux-exo-3

RANCHER

Labels

+ Add Label

ADVANCED OPTIONS ^

Create

Cancel

Now we have a working rancher server running on a cloud instance with a provision of 3 hosts to run containers on them. Let's deploy those 3 containers:

- OpenVPN to create the VPN access.
- Jenkins to build our webapp.
- Tomcat to run our webapp.

## Setup containers on Rancher server

Time to deploy a first service, click on *Add Service* button (you will have to do this add 3 times)

User Stacks

Add Stack

Sort By: State Name

Adding your first Service

A service is simply a group of containers created from the same Docker image but extends Docker's 'link' concept to leverage Rancher's lightweight distributed DNS service for service discovery. Services can be added individually or by deploying an item from the Catalog.

A service is also capable of leveraging other Rancher built-in services such as load balancers, health monitoring, upgrade support, and high-availability. [Learn More](#)

Add Service

Add From Catalog

## OpenVPN

For the OpenVPN container:



1. Scale it to 1 container.

Scale

☒ Run 1 container

☐ Always run one instance of this container on every host

2. Enter a name for it: **rancher-vpn-server**.
3. Enter the docker image: **nixel/rancher-vpn-server:latest**.
4. Add this TCP port map: 1194 (on Host) to 1194 (in Container).
5. Add this TCP port map: 2222 (on Host) to 2222 (in Container).

Name	Description
<input type="text" value="rancher-vpn-server"/>	<input type="text" value="e.g. My Application"/>

Select Image

☒ Always pull image before creating

+ Port Map

Public Host [IP:]Port		Private Container Port		Protocol	
<input type="text" value="1194"/>	>	<input type="text" value="1194"/>	/	TC	-
<input type="text" value="2222"/>	>	<input type="text" value="2222"/>	/	TC	-

+ Service Links

6. In *Volume* section add a new volume to persist the VPN configuration:  
`/etc/openvpn:/etc/openvpn`

Volumes	<div><div>+ /etc/openvpn:/etc/openvpn</div><div>-</div></div>
Volumes From	There are no other launch configs in this service to share volumes with.
Volume Driver	<input type="text" value="Experimental: Requires Docker 1.7"/>

7. In *Security* enable the container *full access to the host* by checking the box.

Privileged	<input checked="" type="checkbox"/> Full access to the host	PID Mode	<input type="checkbox"/> Host
Memory Limit	<input type="text" value="Unlimited"/> <input type="button" value="MB"/>	+ Swap Limit	<input type="text" value="Requires Memory L"/> <input type="button" value="MB"/>
CPU Pinning	<input type="text" value="e.g. 0,3; Default: All"/>	Shares	<input type="text" value="1024"/>
Capabilities	<input type="button" value="Add: None ▾"/>	<input type="button" value="Drop: None ▾"/>	
<small>Capabilities allow you to provide fine-grained control over superuser privileges available to the container. <a href="#">More information</a></small>			
Device Binding	<input type="button" value="⊕"/>		
Log Driver	<input type="text" value="e.g. syslog"/> <input type="button" value="▾"/>		
Log Options	<input type="button" value="⊕ Add Option"/>		

8. And start the container

After a while the container will be ready. But you don't need to wait before creating the other ones.

## Jenkins

1. Scale it to 1 container.

Scale

☒ Run 1 container

☐ Always run one instance of this container on every host

2. Enter a name for it: **jenkins**.

3. Enter the docker image: **jenkins**.

4. No port map is required

Name	Description
<input type="text" value="jenkins"/>	<input type="text" value="e.g. My Application"/>
Select Image	
<input type="text" value="jenkins"/>	
<input checked="" type="checkbox"/> Always pull image before creating	
<input type="button" value="⊕ Port Map"/>	
<input type="button" value="⊕ Service Links"/>	

5. In *Volume* section add a new volume to persist the Jenkins configuration: **/var/jenkins\_home**

Volumes ⊕

/var/jenkins\_home -

Volumes From There are no other launch configs in this service to share volumes with.

Volume Driver Experimental: Requires Docker 1.7

6. And start it!.

## Tomcat

1. Scale it to 1 container.

Scale

☒ Run 1 container

☐ Always run one instance of this container on every host

2. Enter a name for it: **tomcat**.

3. Enter the docker image: **tutum/tomcat:7.0**.

4. No port map is required

Name

tomcat

Description

e.g. My Application

Select Image

tutum/tomcat:7.0

☒ Always pull image before creating

⊕ Port Map

⊕ Service Links

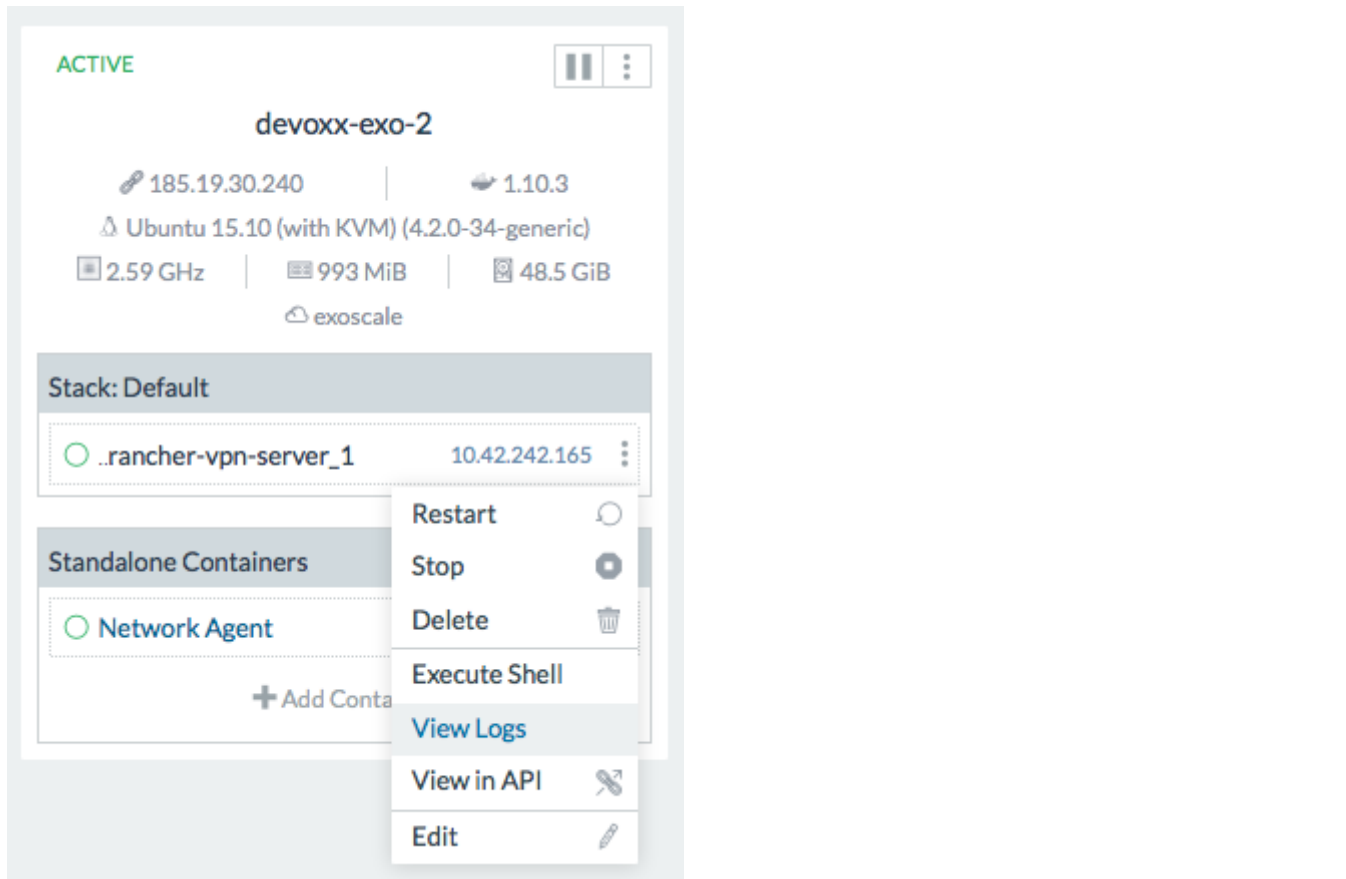
5. And start it!.

Now we have 3 services running on our rancher server:

Stack: <span>Default</span> <span>⌵</span> <span>Add Service</span> <span>⌵</span> <span>⋮</span> <span>⌵</span> <span>⌵</span> <span>⌵</span> <span>Active</span> <span>⌵</span> <span>⋮</span>					
<span>🟢 Active</span>	jenkins ⓘ	Image: jenkins	Service	1 Containers	<span>⌵</span> <span>⋮</span>
<span>🟢 Active</span>	rancher-vpn-server ⓘ	Image: nixel/rancher-vpn-server:latest Ports: 1194, 2222	Service	1 Containers	<span>⌵</span> <span>⋮</span>
<span>🟢 Active</span>	tomcat ⓘ	Image: tutum/tomcat:7.0	Service	1 Containers	<span>⌵</span> <span>⋮</span>

# Connect to the VPN

You need to access the logs of the VPN server to get the generated password to download the configuration. Go in **Infrastructure** → **Hosts** to see the 3 instances details. On the vpn server container, click the 3 vertical dots and choose **View Logs**.



```
03/06/2016 10:17:35 organizationalUnitName:PRINTABLE: 'NIXEL'
03/06/2016 10:17:35 commonName :PRINTABLE: 'RancherVPNClient'
03/06/2016 10:17:35 name :PRINTABLE: 'EasyRSA'
03/06/2016 10:17:35 emailAddress :IASSTRING: 'manel@nixelsolutions.com'
03/06/2016 10:17:35 Certificate is to be certified until Jun 1 08:17:35 2026 GMT (3650 days)
03/06/2016 10:17:35 Write out database with 1 new entries
03/06/2016 10:17:35 Data Base Updated
03/06/2016 10:17:35 /etc/openvpn
03/06/2016 10:17:35 =====
03/06/2016 10:17:35 If you are using nixel/rancher-vpn-client docker image you must run rancher-vpn-client container using the following
03/06/2016 10:17:35 sudo docker run -ti -d --privileged --name rancher-vpn-client -e VPN_SERVERS=185.19.30.240:1194 -e VPN_PASSWORD=Kkd5I
03/06/2016 10:17:35 Then execute "sudo docker logs rancher-vpn-client" so you can view the ip route you need to add in your system in ori
03/06/2016 10:17:35 =====
03/06/2016 10:17:35 If you are using another OpenVPN client (for example for mobile devices) you can get the VPN client configuration exi
03/06/2016 10:17:35 sshpass -p Kkd58ew5gEk0QCcZFrWq ssh -p 2222 -o ConnectTimeout=4 -o UserKnownHostsFile=/dev/null -o StrictHostKeyCheck
03/06/2016 10:17:35 =====
03/06/2016 10:17:35 /usr/lib/python2.7/dist-packages/supervisor/options.py:295: UserWarning: Supervisor is running as root and it is se
03/06/2016 10:17:35 'Supervisor is running as root and it is searching '
03/06/2016 10:17:35 2016-06-03 08:17:35,529 CRIT Supervisor running as root (no user in config file)
03/06/2016 10:17:35 2016-06-03 08:17:35,529 WARN Included extra file "/etc/supervisor/conf.d/supervisord.conf" during parsing
```

Figure 3. Command line in the logs to download the VPN configuration.

## Using a OpenVPN client

Download the configuration file for OpenVPN through SSH as you will see in the log output with the corresponding command line. In my case I got this command:

```
sshpass -p Kkd58ew5gEk0QCcZFrWq ssh -p 2222 -o ConnectTimeout=4 -o
UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no root@185.19.30.240
"get_vpn_client_conf.sh 185.19.30.240:1194" > RancherVPNClient.ovpn
```

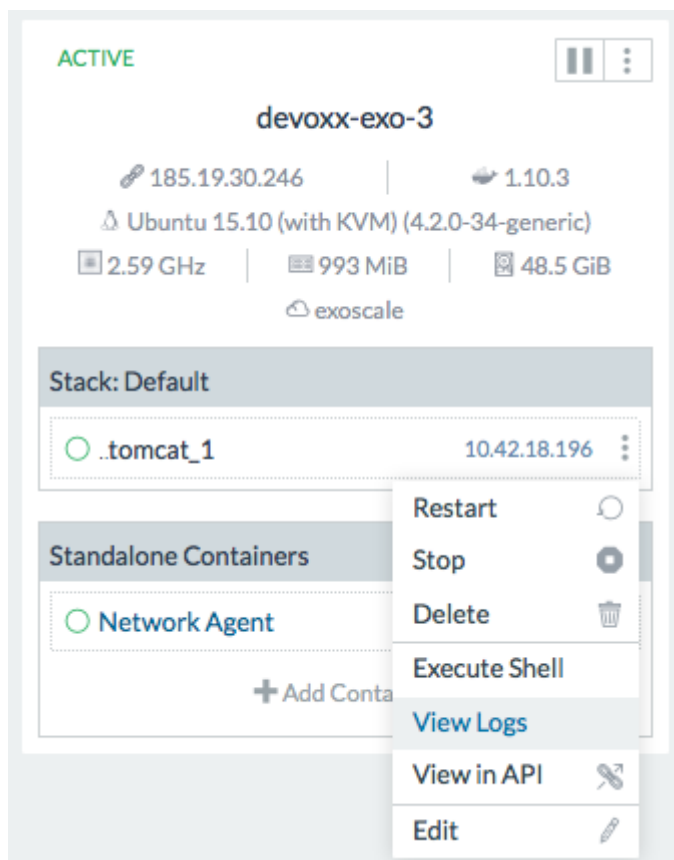
Best is it to do the `ssh -p 2222 ...` command and enter the password in the prompt, skipping the use of `sshpass`.

```
ssh -p 2222 -o ConnectTimeout=4 -o UserKnownHostsFile=/dev/null -o
StrictHostKeyChecking=no root@185.19.30.240 "get_vpn_client_conf.sh
185.19.30.240:1194" > RancherVPNClient.ovpn
# when prompt enter: Kkd58ew5gEk0QCcZFrWq
```

And just load it in your OpenVPN client, and start the connection to the server. Now you are able to connect to Jenkins and tomcat using their own container IP address you'll find on the service view. If you tried before connecting the VPN, you wouldn't have been able.

## Configure Jenkins

First we will need to get the admin password for Tomcat to be able to deploy the app. On the rancher server interface, go to the host view again, and view the Tomcat logs.



## Logs: Default\_tomcat\_1

**Note:** Only combined stdout/stderr logs are available for this container because it was run with the TTY (-t) flag.

**ProTip:** Hold the Command key when opening logs to launch a new window.

```
03/06/2016 15:34:57 ==> Creating an admin user with a random password in Tomcat
03/06/2016 15:34:57 ==> Done!
03/06/2016 15:34:57 =====
03/06/2016 15:34:57 You can now configure to this Tomcat server using:
03/06/2016 15:34:57
03/06/2016 15:34:57      admin:2VPRIHq3Lcq4
03/06/2016 15:34:57
03/06/2016 15:34:57 =====
03/06/2016 15:34:57 Using CATALINA_BASE:   /tomcat
03/06/2016 15:34:57 Using CATALINA_HOME:   /tomcat
03/06/2016 15:34:57 Using CATALINA_TMPDIR: /tomcat/temp
03/06/2016 15:34:57 Using JRE_HOME:        /usr/lib/jvm/java-7-oracle
03/06/2016 15:34:57 Using CLASSPATH:       /tomcat/bin/bootstrap.jar:/tomcat/bin/tomcat-juli.jar
03/06/2016 15:34:59 Jun 03, 2016 1:34:59 PM org.apache.catalina.core.AprLifecycleListener init
03/06/2016 15:34:59 INFO: The APR based Apache Tomcat Native library which allows optimal performance in pro
03/06/2016 15:35:00 Jun 03, 2016 1:35:00 PM org.apache.coyote.AbstractProtocol init
03/06/2016 15:35:00 INFO: Initializing ProtocolHandler ["http-bio-8080"]
```

Figure 4. Keep this admin password for later

Open your browser to the [http://JENKINS\\_CONTAINER:8080](http://JENKINS_CONTAINER:8080) (in my case <http://10.42.45.156:8080/>)

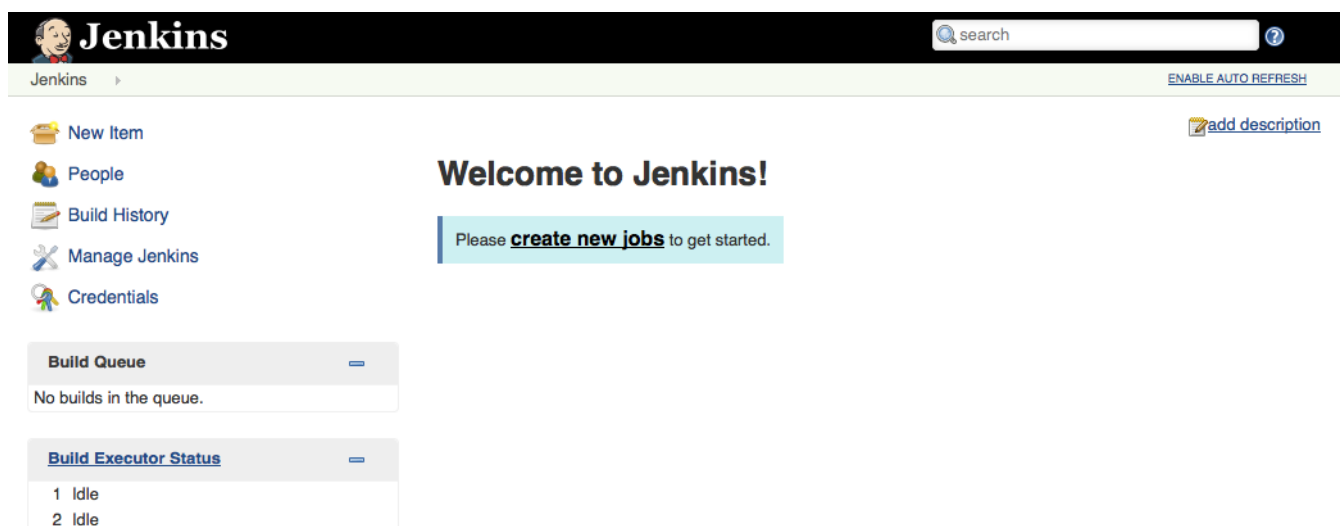


Figure 5. Jenkins Dashboard

Before starting you must install Github Plugin and Maven following these steps:

1. Click *Manage Jenkins* menu option and then *Manage Plugins*
2. Go to *Available* tab and search for Github plugin, named “Github Plugin”. Activate its checkbox
3. Click *Download now and install after restart* button
4. When the plugin is installed enable checkbox *Restart Jenkins when installation is complete and no jobs are running*, and then wait for Jenkins to be restarted
5. When Jenkins is running again, go to *Manage Jenkins* and click *Configure System*
6. In *Maven* section click *Add Maven* button, enter a name for the installation and choose the

maven version you want, in my case the latest available 3.3.9.

## Maven

### Maven installations

#### Maven

Name

☒ Install automatically 

#### Install from Apache

Version  

Delete Installer

Add Installer ▼

Delete Maven

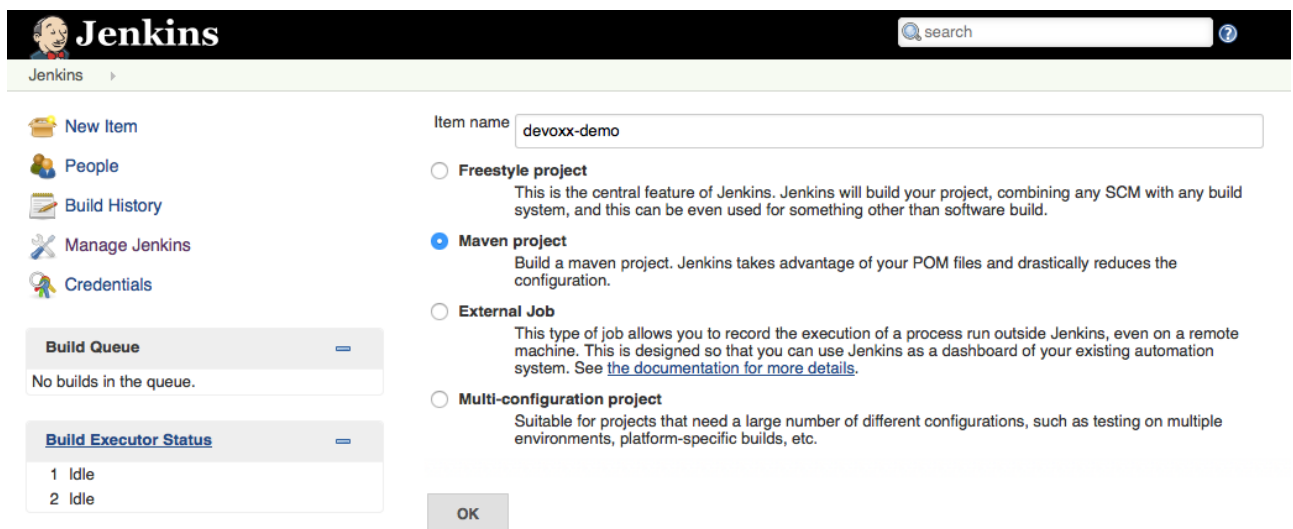
Add Maven

List of Maven installations on this system

7. Click Save button to finish

Create a new job in the *Dashboard*, click *create new jobs* and do as follow:

1. Enter a job name, for example *devoux-demo*
2. Choose *Maven project* and click *OK*



The image shows the Jenkins 'Create new job' dialog. On the left is a sidebar with links: New Item, People, Build History, Manage Jenkins, and Credentials. Below these are two status boxes: 'Build Queue' (empty) and 'Build Executor Status' (showing 1 Idle and 2 Idle executors). The main area has a form with 'Item name' set to 'devoux-demo'. There are four radio button options: 'Freestyle project', 'Maven project' (which is selected), 'External Job', and 'Multi-configuration project'. Each option has a brief description. At the bottom right is an 'OK' button.

3. In *Source Code Management* section choose *Git* and enter this repository url:

- <https://github.com/marcaurele/sample-spring-boot>

4. In *Build* section enter the following maven goals and options. Replace **TOMCAT\_CONTAINER\_IP** with the IP assigned to your Tomcat container (10.42.18.196 in my case) and

**TOMCAT\_ADMIN\_PASSWORD** with the password we saw in the Tomcat logs of the container (2VPRIHq3Lcq4 in my case).

```
clean package tomcat7:redeploy -DTOMCAT_HOST=TOMCAT_CONTAINER_IP -DTOMCAT_PORT=8080
-DTOMCAT_USER=admin -DTOMCAT_PASS=TOMCAT_ADMIN_PASSWORD
# In my case
clean package tomcat7:redeploy -DTOMCAT_HOST=10.42.18.196 -DTOMCAT_PORT=8080
-DTOMCAT_USER=admin -DTOMCAT_PASS=2VPRIHq3Lcq4
```

#### Build

Root POM	<input type="text" value="pom.xml"/>	
Goals and options	<input type="text" value="clean package tomcat7:redeploy -DTOMCAT_HOST=10.42.103.30 -DTOMCAT_PORT"/>	

Advanced...

#### 5. Save the job

Now you can click *Build Now* to run the job. If you check the *Console Output* you will see at the end a *Build success*:

```
[INFO]
[INFO] --- tomcat7-maven-plugin:2.2:redeploy (default-cli) @ sample-spring-boot ---
[INFO] Deploying war to http://10.42.18.196:8080/sample
Uploading: http://10.42.18.196:8080/manager/text/deploy?path=%2Fsample&update=true
Uploaded: http://10.42.18.196:8080/manager/text/deploy?path=%2Fsample&update=true
(12914 KB at 19046.1 KB/sec)

[INFO] tomcatManager status code:200, ReasonPhrase:OK
[INFO] OK - Deployed application at context path /sample
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 37.123 s
[INFO] Finished at: 2016-06-03T13:44:14+00:00
[INFO] Final Memory: 29M/70M
[INFO] -----
[JENKINS] Archiving /var/jenkins_home/jobs/devoxx-demo/workspace/pom.xml to
com.peakxl.demo/sample-spring-boot/1.0.0/sample-spring-boot-1.0.0.pom
[JENKINS] Archiving /var/jenkins_home/jobs/devoxx-demo/workspace/target/sample-spring-
boot-1.0.0.war to com.peakxl.demo/sample-spring-boot/1.0.0/sample-spring-
boot-1.0.0.war
channel stopped
Finished: SUCCESS
```

## Testing the sample app

Now browse to [http://TOMCAT\\_CONTAINER\\_IP:8080/sample/](http://TOMCAT_CONTAINER_IP:8080/sample/) (in my case <http://10.42.18.196:8080/sample/>) and you will see some information about the Tomcat container and your browser.



# Sample SpringBoot app

Server IP addresses:

- fe80:0:0:0:77:d6ff:fe7d:9f90%18
- 10.42.18.196
- 172.17.0.3
- 0:0:0:0:0:0:1%1
- 127.0.0.1

Server Hostname: **f05ddffd11b4**

Your current IP address: **10.42.242.165**

Your current User-Agent: **Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:46.0) Gecko/20100101 Firefox/46.0**

## Deploying the app from your local machine

Of course you can also deploy the app from your local computer to iterate faster by running the maven command locally:

```
git clone https://github.com/marcaurele/sample-spring-boot.git
cd sample-spring-boot
# Replace TOMCAT_CONTAINER_IP and TOMCAT_ADMIN_PASSWORD
mvn clean package tomcat7:redeploy -DTOMCAT_HOST=TOMCAT_CONTAINER_IP
-DTOMCAT_PORT=8080 -DTOMCAT_USER=admin -DTOMCAT_PASS=TOMCAT_ADMIN_PASSWORD

# In my case
mvn clean package tomcat7:redeploy -DTOMCAT_HOST=10.42.18.196 -DTOMCAT_PORT=8080
-DTOMCAT_USER=admin -DTOMCAT_PASS=2VPRIHq3Lcq4
```

```
[INFO] Packaging webapp
[INFO] Assembling webapp [sample-spring-boot] in [/Users/marco/exoscale/sample-spring-boot/target/sample-spring-boot-1.0.0]
[INFO] Processing war project
[INFO] Webapp assembled in [57 msecs]
[INFO] Building war: /Users/marco/exoscale/sample-spring-boot/target/sample-spring-boot-1.0.0.war
[INFO]
[INFO] --- spring-boot-maven-plugin:1.3.5.RELEASE:repackage (default) @ sample-spring-boot ---
[INFO]
[INFO] <<< tomcat7-maven-plugin:2.2:redeploy (default-cli) < package @ sample-spring-boot <<<
[INFO]
[INFO] --- tomcat7-maven-plugin:2.2:redeploy (default-cli) @ sample-spring-boot ---
[INFO] Deploying war to http://10.42.18.196:8080/sample
Uploading: http://10.42.18.196:8080/manager/text/deploy?path=%2Fsample&update=true
Uploaded: http://10.42.18.196:8080/manager/text/deploy?path=%2Fsample&update=true (12914 KB at 7587.1 KB/sec)
[INFO] tomcatManager status code:200, ReasonPhrase:OK
[INFO] OK - Deployed application at context path /sample
[INFO]
[INFO] BUILD SUCCESS
[INFO] Total time: 26.888 s
[INFO] Finished at: 2016-06-03T15:47:02:00
[INFO] Final Memory: 32M/269M
[INFO]
Sample SpringBoot app
sample-spring-boot/ on master
>
```

# Conslusion

We have now a development environment running on a public cloud provider, exoscale, with an encrypted connection to stay safe while coding our app, without to have to think anymore about exposing or mapping ports or editing firewall rules. This setup enables you to work from any location (office, home, wifi hotspot) too, or to give access to your environment to other people.

# Credits

Based on [Manel Martinez Gonzalez](#) post with some modifiations.