Escenic Cloud

Hardware Installation Guide

2.0-SNAPSHOT

escenic

A CCI COMPANY

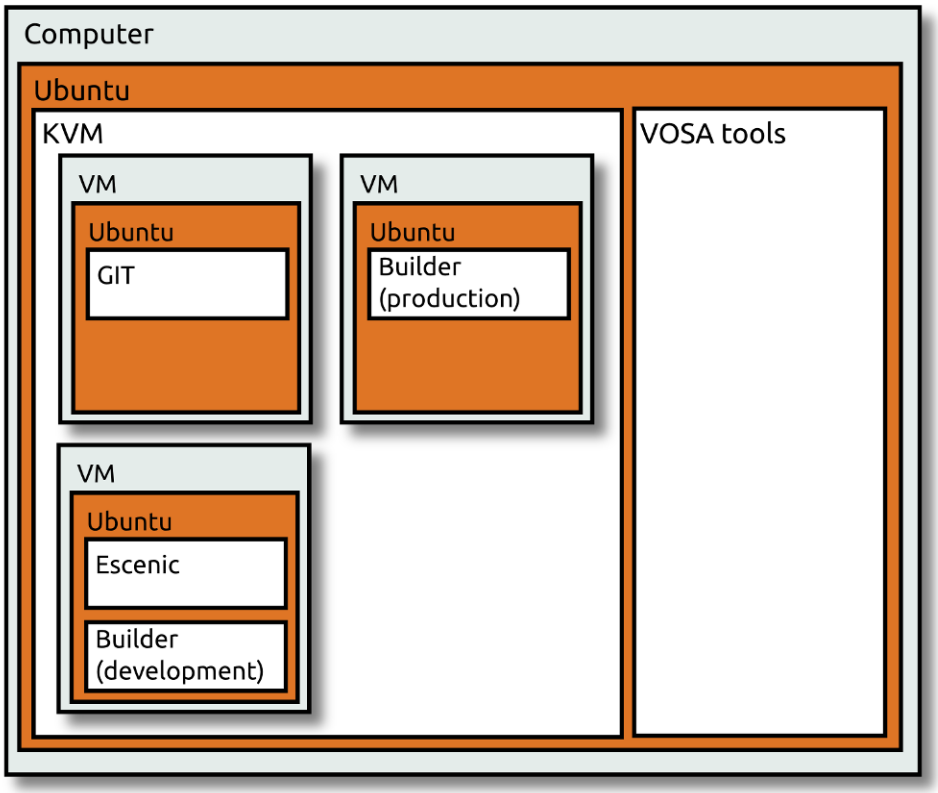
Table of Contents

1 Introduction

Escenic Cloud is a set of tools for installing and configuring an Escenic content management system (CMS) in a cloud environment. "Cloud" in this context is used to mean virtual machines running either on local hardware or on a cloud platform such as Amazon AWS.

The Escenic Cloud tool set consists primarily of bash scripts, and can therefore in principle be installed and used in any Unix-based environment. The recommended operating system platform for Escenic Cloud, however, is Ubuntu — currently Ubuntu 12.04 LTS. All procedures and examples in this manual are therefore based on the use of Ubuntu 12.04 LTS.

This manual describes how to use Escenic Cloud to set up an Escenic CMS running in Ubuntu virtual machines managed by the KVM hypervisor on local hardware. The following diagram shows a minimal Escenic Cloud installation — the installation used as an example in this manual:



In this minimal configuration, the entire Escenic system is installed in one virtual machine; the other two virtual machines are part of the Escenic Cloud infrastructure. One machine contains a **Git repository** for storing application code and another contains the **Escenic Builder**, a set of tools used to build Escenic applications. The installation incorporates two copies of the Escenic Builder:

- A Production Builder, installed in its own VM. This Builder is used to build production versions of Escenic applications.

- A Development Builder, installed alongside the Escenic system. This Builder is used for development purposes. It provides a faster means of generating test builds during application development.

In a production installation, the Escenic system is normally split up and installed on several different virtual machines with specialized functions. This manual does not currently cover installing these more complex system configurations.

This manual also describes how to create a VMWare/VirtualBox image of the Escenic VM you have created, so that it can be distributed to developers as an easily installed, ready-to-use development system.

Throughout this manual, the virtual machines to be created and configured will be referred to by the following names:

`mygit` (the Git VM)
`mybuilder` (the Builder VM)
`mycloud` (the Escenic VM)

1.1 Prerequisites

To create a minimal local Escenic Cloud installation like the one shown in chapter 1, you need a reasonably modern server or PC that meets the following minimum requirements.

- A 64-bit CPU with 8 cores and hardware virtualization support
- 8 GB of RAM (to run one KVM guest)
- Ubuntu 12.04 LTS, 64-bit version

It does not matter whether you use the Desktop or Server Edition of Ubuntu.

1.1.1 Finding out what you've got

If the computer you want to install on already has Ubuntu installed, you can find out exactly what you've got in the way of hardware and software as follows:

1.1.1.1 64-bit processor?

To find out whether your computer has a 64-bit processor or not, enter:

```
egrep -c ' lm ' /proc/cpuinfo
```

A return value of **`0`** means you have a 32-bit CPU. A return value of **`1`** or more means you have a 64-bit CPU.

1.1.1.2 Hardware virtualization supported?

To find out whether or not your processor supports hardware virtualization or not, enter:

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

A return value of `0` means your CPU doesn't support hardware virtualization. A return value of `1` or more means it does support hardware virtualization.

1.1.1.3 Hardware virtualization enabled?

Even if your processor supports hardware virtualization, it may be disabled in the BIOS. You need to make sure that the processor's hardware virtualization extensions are enabled. For a description of how to find out and if necessary enable the extensions, see this page.

1.1.1.4 Ubuntu version?

To find out which version of Ubuntu is installed, enter:

```
lsb_release -a
```

The output from the command will include a **`Description`** line containing the OS version number.

1.1.1.5 64-bit Ubuntu?

To find out whether you are running a 64-bit version of Ubuntu or not, enter:

```
sudo uname --m
```

A return value of **`i686`** means you have a 32-bit version of Ubuntu. A return value of **`x86_64`** means you have a 64-bit version of Ubuntu.

1.2 Credentials

In order to carry out the procedures described in this manual, you need the following credentials. It's a good idea to make sure you have them before you start the process. If you don't have all of these credentials, contact Escenic Support for assistance.

- **Escenic Technet**: user name and password required
- **Escenic Maven Repository**: user name and password required
- **Escenic Debian Repository:** passphrase required

1.3 Conventions

As mentioned in "Prerequisites", you can use either a server or desktop edition of Ubuntu, but all the instructions in this manual assume that you are using a server edition with no X server or graphic capabilities. Therefore all instructions are based on the use of an alphanumeric terminal: either a physical terminal attached to the server, or more likely an SSH session running in a terminal window on another computer. It is assumed that you know how to set up and initiate SSH sessions.

As far as possible, instructions will consist of or include example commands that you can just copy and paste into your terminal. These commands are shown in monospace type like this:

```
ls /home
```

Occasionally you will need to replace parts of the supplied commands with your own values. Such replaceable elements are shown in an italic typeface and explained below if necessary, like this:

```
ls /home/username
```

where *username* is your user name.

It is important that you don't replace supplied filenames or paths except where the italic typeface is used. If you do use names other than the recommended ones, then the resulting installation may not work properly.

Some commands require root privileges. Where this is the case, the commands will be prefixed with **`sudo`**, so that you can just copy and paste these commands too. **`sudo`** will then often prompt for your password, but these prompts are not shown in the instructions.

1.4 Procedure summary

Installing a complete Escenic Cloud system involves the following main tasks:

Set up the host machine
: This involves installing KVM and associated packages, defining a bridge network for the virtual machines, and installing the VOSA toolkit. For details, see chapter 2.

Create the guest virtual machines
: This involves downloading an Ubuntu guest image, defining the characteristics of the VMs to be created, creating the VMs and installing the downloaded image in them. For details, see chapter 3.

Create a Git hosting service
: This involves installing Git and a Git administration tool on one of the guest virtual machines. This step is optional: you can use an existing Git installation if you have one, or a public Git hosting service such as Github or Bitbucket. For details, see chapter 4.

Create an Escenic project
: This involves installing and configuring Maven on your own PC, creating a Git repo for the project, creating the actual project, committing it to the repo and pushing the repo to your Git host. For details, see chapter 5.

Create a Production Builder
: This involves installing a number of SW packages on one of the guest virtual machines followed by a sequence of setup tasks. For details, see chapter 6.

Install Escenic
: This involves creating a couple of system definition files, and then running a single command to install and configure the system. For details, see chapter 7. The Escenic installation will usually include a **Development Builder** (see chapter 8). Once you have a working Escenic installation, you can create copies of the VM for installation on developer PCs using VMWare or VirtualBox (see chapter 9).

1.5 Upgrading

If you already have an Escenic Cloud installation, upgrading to the latest version is very easy. For details, see chapter 10.

2 Set up the host machine

First of all, ensure that your host machine meets the requirements listed in "Prerequisites". If it does, then setting up the host machine involves the following sub-tasks:

- Install KVM and associated packages
- Define a bridge network
- Set up TAP interfaces
- Install the VOSA toolkit

These sub-tasks are described in the following sections.

2.1 Install KVM and associated packages

First of all, make sure your local **apt** database is up-to-date: **sudo apt-get update**

You can then install KVM and all the other tools you need with a single command:

```
sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils uml-utilities
 genisoimage
```

If you need more information regarding KVM installation, you should find it here: https://help.ubuntu.com/community/KVM/Installation

2.2 Define a bridge network

A bridge network allows the virtual machines running in a host computer access to the host computer's network. Once it is set up correctly, the virtual machines become more or less fully functional peers in the host computer's local network. They can access all local resources and the Internet, and are accessible within the network in almost the same way as the host computer.

You can set up a bridge network in both static local networks (where all hosts are assigned fixed IP addresses) and in dynamic DHCP networks. In both cases you set up the bridge by editing the file **/etc/network/interfaces**. You need root access to edit this file, so prefix your editor start-up command with **sudo**.

Once you have made the changes, restart the network subsystem to make them take effect.

```
sudo /etc/init.d/networking restart
```

2.2.1 Static

To set up a bridge network in a static local network add the entries shown in bold to **/etc/network/interfaces**:

```
auto lo
iface lo inet loopback
```

```
auto eth0
iface eth0 inet manual

auto br0
iface br0 inet static
        address
                192.168.0.10
                192.168.0.0
                255.255.255.0
                192.168.0.255
                192.168.0.1
```

Note that the IP addresses and network mask used above are just examples, and should be replaced by suitable addresses for your local network.

2.2.2 DHCP

To set up a bridge network in a local network that is managed by a DHCP server, add the entries shown in bold to **`/etc/network/interfaces`**:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto br0
iface br0 inet dhcp
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
        bridge_maxwait 0
```

Once you have made the changes, restart the network subsystem to make them take effect.

```
sudo /etc/init.d/networking restart
```

2.3 Set up TAP Interfaces

The KVM machine emulator, QEMU, uses TAP interfaces to provide full networking capabilities to its guest operating systems. This necessary in order for:

- Guest operating systems to be able to run several network services connected to via standard ports
- Protocols other than TCP and UDP to be used

The best thing is to set up a number of TAP interfaces that will be available for use if required. The following shell commands, for example, can be run to create 10 TAP interfaces:

```
mkdir -p /var/run/vizrt/vosa
br=br0
for i in $(seq 1 10) ; do
  tap=$(tunctl -b)
  echo $br > /var/run/vizrt/vosa/$tap.availablenetwork
  brctl addif $br $tap
  ifconfig $tap up 0.0.0.0
```

```
done
```

You need to ensure the commands are run every time the machine is rebooted. One way of achieving this is to add them to the **/etc/rc.local** file. Open the file in an editor (you'll need to use **sudo**) and insert the commands at the bottom of the file immediately before the line:

```
exit 0
```

Reboot the machine to execute the commands. If now you enter:

```
ifconfig
```

you should see that all the interfaces **tap1** to **tap10** have been created.

2.4 Install the VOSA Toolkit

VOSA is a toolkit for enabling easily repeatable Escenic Content Engine installations. All the components of an Escenic installation are fully described in an XML definition file called an SDP file. From an SDP file, VOSA can repeatably generate fully working Escenic installations and install them on virtual machines running locally under KVM or in an Amazon cloud. VOSA handles the entire installation process including the download, installation and configuration of all the packages on which Escenic depends.

In addition, VOSA provides a structure for managing the definition and creation of the virtual machines on which Escenic is installed, so that these VMs can also be generated, furnished with an operating system and configured in a repeatable way. VOSA is, in other words, the heart of an Escenic Cloud.

To install VOSA:

1. Add the Escenic SW repository's public key to your host machine's **apt** sources keyring:
   ```
   curl --silent http://apt.escenic.com/repo.key | sudo apt-key add -
   ```
2. Add the Escenic SW repository to the list of source repositories:
   ```
   echo "deb http://apt.escenic.com akita main non-free" | sudo tee -a /etc/apt/
   sources.list
   ```
3. Update your local apt database with the changes:
   ```
   sudo apt-get update
   ```
4. Install VOSA:
   ```
   sudo apt-get install vosa
   ```

3 Create the guest virtual machines

Now you have installed VOSA, you can use it to create the virtual machines you will need. Creating the virtual machines involves the following steps:

1. Create the VOSA configuration folder
2. Download an Ubuntu guest image
3. Create VM definitions
4. Configure the VMs
5. Enable and create the VMs

3.1 Create The VOSA configuration folder

Everything that VOSA does is based on information stored in the VOSA configuration folder — **`/etc/vizrt/vosa/`**.

To create the configuration folder, enter:

```
sudo vosa init
```

The created folder contains the following sub-folders:

`available.d`
Used to hold VM definitions

`enabled.d`
Used to hold symbolic links to **enabled** VM definitions. See section 3.5 for further information.

`overlay`
Used to hold any system files that you want to be copied to the VM after the operating system has been installed. Typically, it contains an **`etc/resolv.conf`** file that you can use to predefine the DNS set-up of your guest VMs. For further information, see section 3.4.2.

`skeleton-amazon`
Used to hold a skeleton VM definition for Amazon AWS clouds installations

`skeleton-kvm`
Used to hold a skeleton VM definition for KVM cloud installations

It also contains one empty file, **`global-ssh-keys`**. This is a global SSH **`known_hosts`** file shared by all the guest VMs you create. Any public key you add to this file will be copied to the **`known_hosts`** files of all your guest VMs.

3.2 Download an Ubuntu guest image

To get the Ubuntu image that will be installed on your guest VMs, enter:

```
sudo vosa -v precise download
```

This command:

1. Downloads the installation package to a subfolder under **/var/lib/vizrt/vosa/uec-images/** (for example, **/var/lib/vizrt/vosa/uec-images/precise-2015-06-03**)
2. Unpacks the downloaded package
3. Creates a symbolic link called **/var/lib/vizrt/vosa/uec-images/current** that points to the new installation folder.

3.3 Create VM definitions

VOSA guest VM definitions are stored in folders under **/etc/vizrt/vosa/available.d**. Each folder contains the definition of one guest VM, and the name of the folder is used as the VM's hostname. This means that the folder names you choose must also be valid host names: they must contain only lowercase letters, numbers and hyphens. Spaces, other punctuation marks and special characters are not allowed.

To create a VM definition, then, you must create a suitably named folder under **/etc/vizrt/vosa/available.d** and then copy the contents of **/etc/vizrt/vosa/skeleton-kvm** into it. To create a host called **mycloud**, for example, you would enter:

```
sudo mkdir /etc/vizrt/vosa/available.d/mycloud
sudo cp /etc/vizrt/vosa/skeleton-kvm/* /etc/vizrt/vosa/available.d/mycloud/
```

You can repeat this process to create as many VM definitions as you require. In order to create the system illustrated in chapter 1 you need to create three such definitions:

- **mygit** to host Git (see chapter 4)
- **mybuilder** to host an Escenic Cloud Production Builder (see chapter 6)
- **mycloud** to host an Escenic installation (see chapter 7)

The full sequence of commands to create all three VMs, then, is:

```
sudo mkdir /etc/vizrt/vosa/available.d/mygit
sudo cp /etc/vizrt/vosa/skeleton-kvm/* /etc/vizrt/vosa/available.d/mygit/
sudo mkdir /etc/vizrt/vosa/available.d/mybuilder
sudo cp /etc/vizrt/vosa/skeleton-kvm/* /etc/vizrt/vosa/available.d/mybuilder/
sudo mkdir /etc/vizrt/vosa/available.d/mycloud
sudo cp /etc/vizrt/vosa/skeleton-kvm/* /etc/vizrt/vosa/available.d/mycloud/
```

3.4 Configure the VMs

A VM definition copied from **/etc/vizrt/vosa/skeleton-kvm/** consists of two configuration files called **boot.conf** and **install.conf**. You need to edit both of them and make sure that all the configuration parameters they contain are correctly set. You need root access to edit the files, so prefix your editor start-up command with **sudo**.

3.4.1 Editing boot.conf

`boot.conf` contains parameters that govern the environment in which a VM will run — the physical resources it will have access to. The parameters are:

3.4.1.1 memory

The amount of memory to be allocated to the VM, specified in Mb. The following setting, for example, allocates 3.5 Gb to the VM:

```
memory 3584
```

The recommended memory allocations for the basic development installation described in this manual are:

VM usage	Recommended	Minimum viable
Git server	3.5	2
Development Builder	3.5	2
Escenic all-in-one	8	3.5

Bear in mind the following factors:

1. You need to leave 4 GB of memory for the KVM host machine, so you can't allocate everything to your guest VMs.
2. KVM does allow you to overcommit memory, but you should not overcommit too much, and you must ensure that the KVM host machine has enough swap space to cope with it.
3. If you know that certain VMs will never be run simultaneously, then you can exclude those combinations from your allocation calculations. This is not the case here: all three VMs are intended to run simultaneously.

For a detailed description of overcommitting and KVM, see https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Administration_Guide/chap-Virtualization-Tips_and_tricks-Overcommitting_with_KVM.html.

3.4.1.2 processor_affinity

The host machine processors that the guest VM will have access to. The default setting is:

```
processor_affinity 0,1,2,3
```

which gives the VM access to the host's first four processors. How you set this parameter depends on how many processors your host computer has, how many guest VMs you are installing, and what their workloads are. Several VMs can share the same processors without any problem (other than performance problems if they overload the processors), but you may wish to allocate different processors to different VMs in order to spread the workload as evenly as possible between the available processors.

3.4.1.3 bridge

The name of the bridge network that will be used to access the guest VMs. This should be the name you used when defining the bridge network, as described in section 2.2:

```
bridge br0
```

3.4.2 Editing install.conf

`install.conf` contains parameters that determine the operating system to be installed in the guest VM, and the resources that the guest VM will make available to the operating system.

original_image, kernel, initrd

These parameters determine which image is installed in the guest VM. The default settings point to the Ubuntu cloud image you downloaded earlier, and you should not modify them:

```
original_image /var/lib/vizrt/vosa/uec-images/current/disk.img
kernel /var/lib/vizrt/vosa/uec-images/current/vmlinuz
initrd /var/lib/vizrt/vosa/uec-images/current/initrd
```

initial_disk_size

The size of the guest VM's disk, specified in Gb. For example:

```
initial_disk_size 10
```

When a VM is initialized, its disk is created by copying the disk image specified with the **`original_image`** parameter. It is then resized to the size specified with this parameter. If you don't specify this parameter then the disk will not be resized and not have any space for additional content.

So you should set this parameter to a value that is large enough to cope with your expected requirements. The default value of 10 GB is fine for test and experimental purposes.

timezone

The time zone in which the guest VM is to run, for example:

```
timezone Europe/Oslo
```

For a list of valid time zone names, see http://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

overlay

The path of a folder in your VOSA configuration containing files that you want to be copied to the guest VM during the OS installation process. You can, for example, preconfigure the guest VM's DNS set-up by creating an overlay folder in your VOSA configuration that contains an **`etc/resolv.conf`** file with your preferred settings. All the files found in the specified overlay folder are copied to the root of the guest VM. **`/etc/vizrt/vosa/overlay/common/etc/resolv.conf`**, for example, will be copied to **`/etc/resolv.conf`** in the guest VM.

The default setting is:

```
overlay /etc/vizrt/vosa/overlay/common
```

macaddr

The MAC address to be assigned to the guest machine. For example:

```
macaddr try with changing only the last 6 digit.00:0c:40:62:4f:2c
```

There are various recipes for generating MAC addresses on Linux systems to be found on the Internet. Here is one of them.

ip_address, netmask, gateway

The guest VM's network configuration parameters, for example:

```
ip_address 192.168.0.221
netmask 255.255.255.0
gateway 192.168.0.1
```

Note that the IP addresses and network mask used above are just examples, and should be replaced by suitable addresses for your local network. The **`netmask`** and **`gateway`** parameters will usually be the same as the **`netmask`** and **`gateway`** parameters you set for the bridge network in **`/etc/network/interfaces`** (see section 2.2).

mirror

The Ubuntu mirror from which the guest VM is to download system packages and updates. For example:

```
mirror http://no.archive.ubuntu.com/ubuntu
```

In order to ensure fast, reliable downloads, set it to point to a mirror in your region. For a list of available Ubuntu mirrors, see here.

postinstall

You can use this parameter to specify Shell scripts that are to be run after the OS has been installed. You can specify the parameter as many times as you like in order to run multiple scripts. The scripts are executed in the order they are specified.

This parameter only has any effect for VMs on which you are going to install Escenic software using the VOSA **`install`** command (see chapter 7).

For a VM on which you intend to install Escenic SW with VOSA, you must specify at least the following scripts:

```
postinstall wait-for-ssh.sh
postinstall make-a-motd.sh
postinstall run-sdp-bootstrapper.sh
```

`wait-for-ssh.sh` waits until the SSH server is working on the guest VM, and then exits. You should **always** specify this script first — it ensures that SSH is available to be used by the following scripts.

`make-a-motd.sh` puts a "message of the day" in the guest VM's **`/etc/motd`** file.

`run-sdp-bootstrapper.sh` starts the process of installing an Escenic system.

In addition to these supplied scripts, you can specify custom scripts you have made yourself. For more about this, see section 3.4.2.1.

3.4.2.1 Custom post-install scripts

The recommended way of running a custom script is to create a **hooks** folder in the VM definition folder (**/etc/vizrt/vosa/available.d/***vm-name***/hooks**, for example) and place the script there. You can then specify the script as follows:

```
postinstall ./hooks/script-name
```

Custom scripts are executed by the VOSA tool on the KVM host, not on the guest VM. In order to perform actions on the guest VM they need to make use of SSH-based tools such as **scp** and **ssh**.

The following example script **sayhello.sh** is intended to create a file called **/tmp/test/hello** on the guest VM:

```
#!/bin/bash -e
echo "--- START : post install hook - create a test file---"
ssh -F $2/ssh.conf root@guest 'mkdir /tmp/test'
ssh -F $2/ssh.conf root@guest 'echo "Hello World! " >> /tmp/test/hello'
```

In order to use this script, you need to:

1. Save it in **/etc/vizrt/vosa/available.d/** *vm-name* **/hooks**
2. Make it executable:
   ```
   chmod +x /etc/vizrt/vosa/available.d/vm-name/hooks/sayhello.sh
   ```
3. Add a **postinstall** entry for the script to **install.conf**:
   ```
   postinstall ./hooks/sayhello.sh
   ```
   Make sure that you add the entry **after** the **postinstall wait-for-ssh.sh** entry.

sayhello.sh will now be run after installation of the VM.

You can also rerun **postinstall** hooks manually at any time after the VM has been installed, as long as you pass in the following two parameters:

1. The path of the VM definition folder (**/etc/vizrt/vosa/available.d/***vm-name*, for example)
2. The system folder in which VOSA has stored the VM's SSH configuration details (**/var/lib/vizrt/vosa/images/***vm-name*, for example)

To run **sayhello.sh** on a VM called **mycloud** you would need to enter:

```
cd /etc/vizrt/vosa/available.d/mycloud
./hook/sayhello.sh . /var/lib/vizrt/vosa/images/mycloud
```

3.5 Enable and create the VMs

To enable a VM, enter the following command:

```
sudo vosa -i host-name enable
```

where *host-name* is the name of your guest VM. For example:

```
sudo vosa -i mycloud enable
```

All that this does is create a symbolic link with the path **`/etc/vizrt/vosa/enable.d/mycloud`** that points to your **`/etc/vizrt/vosa/available.d/mycloud/`** folder.

Once you have enabled your VMs you can finally create them. To create a VM enter the following command:

```
sudo vosa -i host-name create
```

where *host-name* is the name of your guest VM. For example:

```
sudo vosa -i mycloud create
```

You can then check the status of the machine by entering:

```
sudo vosa -i mycloud status
```

The full sequence of commands required to enable and create all three VMs illustrated in chapter 1 is:

```
sudo vosa -i mygit enable
sudo vosa -i mygit create
sudo vosa -i mybuilder enable
sudo vosa -i mybuilder create
sudo vosa -i mycloud enable
sudo vosa -i mycloud create
```

4 Create a Git hosting service

All your development code should be kept in a version control system, and Escenic recommends the use of GIT. You can, if you wish use a public Git hosting service such as Github or Bitbucket, or if you already have an in-house Git hosting service, you can use that. Alternatively you can set up a Git hosting service.

This chapter describes how to set up a Git hosting service in one of your Escenic Cloud virtual machines. The description assumes that you install it on its own in a dedicated VM, but you could also install it in a shared VM together with other Escenic Cloud components.

The Git hosting service is provided by a tool called **gitolite**. Gitolite is a lightweight Git hosting service that provides repository administration via **`gitolite-admin`**. **`gitolite-admin`** is itself a Git repository that is cloned to the administrator's PC. The administrator can then carry out many administration tasks (such as the creation of repositories and users) by simply making changes to files in his **`gitolite-admin`** repository and pushing them to the Git host.

Setting up the Git repository involves the following steps:

1. Log in to the VM you want to use for Git hosting
2. Install Git and Gitolite
3. Create users
4. Setup Gitolite
5. Setup the Gitolite administrator's working environment

Once you have completed these tasks, you can start creating Git users and repositories.

4.1 Log in to your Git hosting VM

To log in to the Git virtual machine:

1. Log in to the KVM host machine.
2. Use SSH to log in to the required guest VM. To do this you need to supply the path of the VM's SSH configuration file (**`ssh.conf`**) stored in **`/var/lib/vizrt/vosa/images/`** *vm-name*. If your Git VM is called **`mygit`**, for example, then you should use the following command to log in:

   ```
   sudo ssh -F /var/lib/vizrt/vosa/images/mygit/ssh.conf root@guest
   ```

4.2 Install Git and Gitolite

Once you have logged in, install the Git and Gitolite packages as follows:

```
apt-get install git-core
apt-get install gitolite
```

4.3 Create a Git user

You now need to create a user on the Git VM to act as the owner of the Git and Gitolite installations. You can call the user **`git`**.

To create the **`git`** user, enter the following command:

```
adduser --system --shell /bin/bash --group --disabled-password --home /home/git git
```

4.4 Setup Gitolite

You initialize Gitolite by running a setup program called **`gl-setup`**, to which you must supply the SSH public key of the user who is to be the Gitolite administrator. The Gitolite administrator is responsible for creating and deleting Git repositories and users, and for controlling user access rights.

The first setup task, therefore, is to copy the administrator's SSH public key to the Git VM. Assuming that:

1. You are the administrator
2. You have already generated an SSH key pair on your own PC
3. You have SSH access from your own PC to the KVM host machine

then you can copy your SSH public key to the Git VM as follows:

1. **On your own PC**, copy your key to the KVM host machine by entering:

   ```
   scp ~/.ssh/id_rsa.pub user@kvm-host:/tmp/$(whoami).pub
   ```

   where *kvm-host* is the host name or IP address of your KVM host machine and *user* is your user name on the KVM host machine. This copies the key to **`/tmp/`***your-user-name*`.pub`** on the KVM host (where *your-user-name* is your user name on your own PC).
2. **On the KVM host machine**, copy the key to the Git VM by entering:

   ```
   sudo scp -F /var/lib/vizrt/vosa/images/mygit/ssh.conf /tmp/your-user-name.pub
    root@guest:/tmp/
   ```

Note that the name of the **`.pub`** file you copy to the Git VM will be used by Gitolite as the administrator user name.

With the administrator's public key in place, you can now finish the setup task as follows:

1. From the KVM host machine, log in to the Git VM:

   ```
   sudo ssh -F /var/lib/vizrt/vosa/images/mygit/ssh.conf root@guest
   ```
2. Change user to **`git`**:

   ```
   su - git
   ```
3. Run the Gitolite setup command, supplying your public key:

   ```
   gl-setup /tmp/your-user-name.pub
   ```
4. **`gl-setup`** creates a Gitolite configuration file and opens it in the **`vi`** editor so that you can make changes if you wish. However, no changes are required at this stage, so you can just quit the editor by entering **`:q`**.

4.5 Set up the Gitolite administrator's working environment

Gitolite stores all the Git administration data in a special Git repository to which only the administrator has access. Setting up the administrator's working environment, therefore, is simply a matter of setting up a normal Git working environment and then cloning the admin repository.

So, assuming still that you are the Gitolite administrator:

1. Go back to your own PC and if necessary, install Git.
2. Create or CD to a folder where you want to keep your Git repositories.
3. If you have not already done so, configure Git by supplying a user name and email address:

   ```
   git config --global user.email "you@example.com"
   git config --global user.name "Your Name"
   ```

   (This is not related to Gitolite administration, it's a general Git setup task.)
4. Clone the Gitolite admin repository as follows:

   ```
   git clone git@ip-address:gitolite-admin.git
   ```

   where *ip-address* is the IP address of your Git VM - the one you specified when defining the VM as described in section 3.4.2.

4.6 Gitolite administration quick start

You are now the Gitolite administrator. You will see that the **`git clone`** command you just executed has created a subfolder called **`gitolite-admin`**. If you look in the folder you will see that it contains the following structure:

`gitolite-admin`
  `conf`
    `gitolite.conf`
  `keydir`
    your-user-name`.pub`

To add users:

1. Get the users' SSH public key files.
2. Copy them into the **`gitolite-admin/keydir`** folder and rename them to *user-name*`.pub`.
3. Commit and push the changes back to the Git host:

   ```
   git add keydir/*.pub
   git commit -am "Added new users"
   git push -u origin master
   ```

To add repositories (or **repos**, as they are more commonly known):

1. Open **`gitolite-admin/conf/gitolite.conf`** in an editor. You will see the following:

   ```
   repo gitolite-admin
        RW+   =   your-user-name
   repo testing
        RW+   =   @all
   ```

2. To create a new repo just add a similar pair of lines to the end of the file:

```
repo new-repo-name
     RW+   =   your-user-name
```

   If you want other users to be able to access the repo, add more access specifications, for example:

```
repo new-repo-name
     RW+   =   your-user-name
     R     =   other-user-name
```

For a complete description of Gitolite and how to use it, see the Gitolite web site at http://gitolite.com/.

5 Create an Escenic project

Escenic Cloud applications are built using Maven, and must conform to a tightly-specified structure in order to work. In order to simplify the process of creating this structure, Escenic Cloud includes a Maven archetype (that is a project template) from which you can generate a correctly structured Maven project. Once you have created a working project it must be committed to a Git repo and pushed to your hosting service.

The full procedure is as follows:

1. Install Maven on your own PC (if it isn't already installed).
2. Configure Maven to be able to access the Escenic Maven repository, so that it has access to all the necessary Escenic components.
3. Create a Git repo for the project
4. Create the project
5. Commit the project and push it to your Git host

5.1 Install Maven

Install the current stable version of Maven, which you can get from http://maven.apache.org/download.cgi. If you use Linux, then you should check to see if you can install it from your distribution's repositories first.

5.2 Configure Maven

You configure Maven by editing its settings file, called **`settings.xml`**. For general information about this file, see http://maven.apache.org/settings.html. The specific settings you need to add are a repository definition something like this:

```
<repositories>
  …
  <repository>
    <id>escenic-repo</id>
    <name>Escenic</name>
      <url>http://maven.escenic.com</url>
    <layout>default</layout>
  </repository>
  …
</repositories>
```

and a corresponding set of credentials for accessing the repository:

```
<server>
  <id>escenic-repo</id>
  <username>username</username>
  <password>password</password>
</server>
```

where *username* and *password* are the credentials you have been supplied with by Escenic. If you don't have any Maven credentials, contact Escenic Support for assistance.

5.3 Create a Git repo

1. Go to the **gitolite-admin** folder on your PC.
2. Open **conf/gitolite.conf** for editing.
3. Add the following lines:

```
repo mycloud
     RW+   =   your-user-name
```

4. Save your changes.
5. Commit and push the changes back to the Git host:

```
git add conf/gitolite.conf
git commit -am "Added mycloud repository"
git push -u origin master
```

The response from Git will include this warning:

```
You appear to have cloned an empty repository.
```

but that is OK, that's what you want. You should now have a **mycloud** folder alongside your **gitolite-admin** folder, and the next step is to fill it by creating an Escenic project.

5.4 Create the project

In your main Git repo folder, enter the following command:

```
mvn -Pdevelopment archetype:generate \
-DarchetypeGroupId=com.escenic \
-DarchetypeArtifactId=saas-archetype \
-DarchetypeVersion=5.7-release-SNAPSHOT \
-DgroupId=your-group-id \
-DartifactId=mycloud \
-DinteractiveMode=false
```

You will see that executing this command fills the **mycloud** folder with a complete Escenic project structure. The project structure is based on **saas-archetype**, a Maven **archetype** (or project template) downloaded from the Escenic Maven repository. The version number **5.7-release-SNAPSHOT** ensures that the project is based on the latest available release of Escenic 5.7. You can specify whatever group ID you want. It should be a dotted string based on your organization name - something like **com.***yourcompany***.test** or **com.***yourcompany***.anythingyoulike**. You must use the name of your project (**mycloud**) as your **artifactId**.

5.5 Commit and push

The project is created. All you need to do now is commit it and push it to your Git host:

```
cd mycloud
git add *
git commit -m "Adding cloud test project"
git push -u origin master
```

6 Create the Production Builder

In addition to all the base components of an Escenic system, a working Escenic installation consists of a number of web applications that must be deployed on a web application server (Tomcat). These include both base applications supplied as part of the delivered system (Web Studio and Mobile Studio, for example), and user applications such as publications.

These applications must packaged together in an **enterprise archive** or **ear file** in order to be deployed. This packaging process is called **assembly** and is performed by a script called the Escenic **assembly tool**. It is possible to manually assemble and deploy an Escenic system using only the assembly tool. Escenic Cloud, however, also provides a set of build scripts that greatly simplify the assembly and deployment process. This collection of tools, once it is correctly installed and configured, is called an Escenic **Builder**.

There are two main builder types:

1. A **Production Builder** is usually installed on its own, either in a guest VM or in the KVM host. It builds from code checked into the master Git repo on the Git VM and produces EAR files ready for deployment on all production Escenic hosts.
2. A **Development Builder** is installed in a guest VM along with an all-in-one Escenic installation. It builds from the developer's local Git repo and automatically deploy the result to the local host. This provides a much more streamlined and efficient workflow for developers.

This chapter describes how to install and configure the Production Builder in a guest VM called **`mybuilder`**.

> Even if you only intend to use your Escenic Cloud installation for development purposes, and have no immediate need for a Production Builder, you still need to install it. If you don't first install a Production Builder in the **`mybuilder`** VM then you will not be able to install Escenic in your **`mycloud`** VM.

There are two different types of Production Builder:

1. **Multi-Project Builder**: This kind of builder is able to support multiple development teams working on different sets of publications, and is usually only required for very large Escenic installations
2. **Single-Project Builder**: This kind of builder is suitable for smaller installations where a single development team is responsible for developing a single publication or group of related publications.

The Single-project Builder is easier to install, so if you are sure you won't need multi-project support, that is the option you should choose.

6.1 Multi-Project Builder

Creating a Multi-Project Builder involves the following steps:

1. Log in to the **`mybuilder`** VM

2. Install various required software packages
3. Initialize the builder
4. Create one or more customer accounts
5. Configure the customer account(s)
6. Enable Git access
7. Configure Maven
8. Install and configure the Escenic Assembly Tool
9. Make a preliminary build
10. Add missing configuration layers
11. Rebuild the project
12. Configure a web server

6.1.1 Log in to your Builder VM

To log in to your builder virtual machine:

1. Log in to the KVM host machine.
2. Use SSH to log in to the required guest VM. To do this you need to supply the path of the VM's SSH configuration file (**`ssh.conf`**) stored in **`/var/lib/vizrt/vosa/images/`** *vm-name*. If your builder VM is called **`mybuilder`**, for example, then you should use the following command to log in:

```
sudo ssh -F /var/lib/vizrt/vosa/images/mybuilder/ssh.conf root@guest
```

6.1.2 Install required software packages

All of the software packages needed on the Builder can be installed from **`apt`** repositories. It's a good idea, however, to install Maven manually, as the standard Maven **`apt`** package for Ubuntu contains a number of additional packages that are not needed.

So you need to:

1. Install the Oracle JDK using **`apt-get`**
2. Add the Escenic **`apt`** repository
3. Install all other **`apt`** packages
4. Manually install Maven

6.1.2.1 Install the Oracle JDK

Installing the Oracle JDK using **`apt-get`** is a slightly more complicated procedure than for most packages, because you have to agree to the Oracle license during the installation procedure. Enter the following commands:

```
apt-get install python-software-properties
add-apt-repository ppa:webupd8team/java
apt-get update
apt-get install oracle-java8-installer
```

The last command displays a license agreement that you need to agree to, using a character-based UI. Use your tab key to move the selection highlight between options, and press the spacebar to select them.

Verify that the JDK is installed and that you have the latest version by entering:

```
java -version
```

6.1.2.2 Add the Escenic apt repository

Add the Escenic SW repository as follows:

1. Add the Escenic repository's public key to your VM's **`apt`** sources keyring:
   ```
   curl --silent http://apt.escenic.com/repo.key | apt-key add -
   ```
2. Add the Escenic repository to the list of source repositories:
   ```
   echo "deb http://apt.escenic.com akita main non-free" | tee -a /etc/apt/
   sources.list
   ```
3. Update your local **`apt`** database with the changes:
   ```
   apt-get update
   ```

6.1.2.3 Install other apt packages

All the remaining **`apt`** packages can be installed with a single command:

```
apt-get install escenic-common-scripts escenic-build-scripts ant zip unzip subversion
 git-core xml2 alien fakeroot nginx
```

6.1.2.4 Install Maven

To install Maven, enter the following commands:

```
cd /tmp
wget http://www.us.apache.org/dist/maven/maven-3/3.3.3/binaries/apache-maven-3.3.3-
bin.zip
unzip apache-maven-3.3.3-bin.zip -d /opt/
mv /opt/apache-maven-3.3.3 /opt/apache-maven
```

> The download URL given above (**`http://www.us.apache.org/dist`**) is an Apache backup mirror. You can probably improve the download speed by choosing a local mirror instead. For a list of official download mirrors, look here.

6.1.3 Initialize the Builder

To initialize the Builder:

1. Enter the following command:
   ```
   ece-builder initialize
   ```
   This generates two configuration files in the **`/etc/escenic`** folder:
   ```
   /etc/escenic/builder.conf
   /etc/escenic/user.conf
   ```
2. Open **`/etc/escenic/builder.conf`** in an editor, and edit the following four lines:

```
technet_user=technet-user
technet_password=technet-pass
maven_vizrt_user=maven-user
maven_vizrt_password=maven-pass
```

Replace *technet-user*, *technet-pass*, *maven-user* and *maven-pass* with your Technet and Escenic Maven repository credentials. If you don't have these credentials, contact Escenic support.

3. Save your changes and then initialize the builder by entering the following command:

```
ece-builder -i /etc/escenic/builder.conf
```

6.1.4 Create customer accounts

A builder **customer account** is intended to represent an organization or organizational unit responsible for developing and maintaining a set of one or more related Escenic projects. A separate Linux user is created for each customer account, and each customer account will have its own Git account.

You must create at least one customer account, and in our case it should be called **`mycloud`**.

To create a customer account:

1. Open **`/etc/escenic/user.conf`** in an editor:

```
user_name=mycloud
user_svn_path=https://vizrtcustomers.jira.com/svn/mycloud/
user_svn_username=mycloud-build-user
user_svn_password=mycloud-password
user_maven_username=mycloud-dev
user_maven_password=adjective-noun
```

2. Set the parameters in the file as follows:

   `user_name`
   The name of the customer account, **`mycloud`** in this case.

   `user_svn_path`, **`user_svn_username`**, **`user_svn_password`**
   All of these parameters are now redundant, but still required by the script. Just set them to the values shown above.

   `user_maven_username`
   The Maven user name must be formed by appending **`-dev`** to the specified **`user_name`**.

   `user_maven_password`
   You should try to choose a good but reasonably memorable password. We recommend an *adjective-noun* combination (**`inglorious-developer`**, for example). You can get such combinations from a generator at http://watchout4snakes.com/wo4snakes/Random/RandomPhrase.

3. Save the file.
4. Enter the following command:

```
ece-builder -u /etc/escenic/user.conf
```

If you require additional customer accounts, repeat the procedure described in this section, specifying a different account name each time.

6.1.5 Configure the customer accounts

Linux users have now been created for your customer accounts. To configure a customer account you need to:

1. Switch to the new user:

   ```
   su - mycloud
   ```

2. Open the user's **~/.build/build.conf** file for editing.
3. Delete the existing contents of **build.conf**:

   ```
   customer=mycloud
   svn_base=https://vizrtcustomers.jira.com/svn/mycloud/
   svn_user=mycloud-build-user
   svn_password=mycloud-password
   ```

4. And replace them with the following:

   ```
   customer=mycloud
   src_control=git
   git_user=git
   git_protocol=ssh
   git_base=git-ip-address/mycloud.git
   ear_base_url=http://builder-ip-address:81
   java_8_oracle=/usr/lib/jvm/java-8-oracle
   ```

   where:

   - *git-ip-address* is the IP address of your Git VM
   - *builder-ip-address* is the IP address of your Builder VM (that is, the VM you are currently working on)

   Don't forget to append the port number **:81** to the end of **ear_base_url**.

5. Save your changes.

> If you have created additional customer accounts, repeat the procedure described in this section, specifying a different account name each time.

6.1.6 Enable Git access

The **mycloud** user needs access to Git in order to be able to build your project. So you need to:

1. Create an SSH key pair for the user
2. Add the SSH public key to the list of Git users using Gitolite on the Git VM, and grant the user read access to the **mycloud** repo

In detail, the procedure is as follows:

1. Enter the following command to create a key pair:

   ```
   ssh-keygen -t rsa -b 4096 -C "your-email@example.com"
   ```

   ssh-keygen outputs three prompts: you should respond to all of them by pressing **Enter**:

   ```
   Enter file in which to save the key (/home/mycloud/.ssh/id_rsa): [Press enter]
   Enter passphrase (empty for no passphrase): [Press enter]
   Enter same passphrase again:  [Press enter]
   ```

It is particularly important that you do not specify a passphrase, as the builder needs to be able to access Git without any human intervention.

After you have responded to these prompts, the keys are generated and **ssh-keygen** finishes by outputting a fingerprint of the public key, for example:

```
The key fingerprint is:
a2:53:50:1d:c7:7e:c6:00:e8:a2:ca:c0:cb:d9:cf:f1 your-email@example.com
```

You do not need to use this fingerprint for anything.

2. List the content of **/home/mycloud/.ssh/id_rsa.pub** by entering

```
cat /home/mycloud/.ssh/id_rsa.pub
```

3. Select the listed content and copy it. Make sure you copy all of the content, including ssh-rsa at the beginning and the email address at the end.
4. Close the file again.
5. Back on your own PC, create an empty file called **mycloud.pub** in your **gitolite-admin/keydir** folder.
6. Paste the text you copied into the file and save it.
7. Open **conf/gitolite.conf** for editing.
8. Grant the **mycloud** user read-only access to the mycloud repo:

```
repo mycloud
     RW+ = your-user-name
     R   = mycloud
```

9. Commit and push the changes back to the Git host:

```
git add keydir/mycloud.pub
git add conf/gitolite.conf
git commit -m "Added mycloud build user"
git push -u origin master
```

If you have created additional customer accounts/users, repeat the procedure described in this section, specifying a different account name each time.

6.1.7 Configure Maven

To set up Maven correctly, return to the **mybuilder** VM and:

1. Make sure you are logged in as the **mycloud** user - if not, enter:

```
su - mycloud
```

2. Open **/home/mycloud/.bashrc** in an editor.
3. Add the following lines at the bottom of the file:

```
export PATH=$PATH:/opt/apache-maven/bin
export M2_HOME=/opt/apache-maven
```

4. Save and close the file.
5. Check that Maven is correctly installed by entering:

```
source /home/mycloud/.bashrc
mvn -version
```

This should produce a few lines of version information from Maven.

6. Open **/home/mycloud/.m2/settings.xml** in an editor.
7. Add your credentials for accessing the Escenic repository by replacing the whole of the existing **servers** element with:

```
<servers>
  <server>
    <id>vizrt-repo</id>
    <username>username</username>
    <password>password</password>
  </server>
  <server>
    <id>vizrt-unstable</id>
    <username>username</username>
    <password>password</password>
  </server>
</servers>
```

   where *username* and *password* are the credentials you have been supplied with by Escenic. If you don't have any Maven credentials, contact Escenic Support for assistance. Note that you need to enter the credentials twice, as shown above: once for the stable **escenic-repo** and once for **escenic-unstable**.

If you have created additional customer accounts/users, repeat the procedure described in this section for each user.

6.1.8 Install and configure the Escenic assembly tool

1. Go to the **mycloud** user's home folder, and download the assembly tool from Escenic Technet with the following command:

```
cd /home/mycloud
wget http://user:password@technet.escenic.com/downloads/release/57/
assemblytool-2.0.7.zip
```

   where *user* and *password* are replaced with your Escenic Technet credentials. If you do not have the necessary credentials, please contact Escenic Support.
2. Unzip the downloaded file and initialize the assembly tool:

```
unzip assemblytool-2.0.7.zip -d assemblytool-2.0.7
cd assemblytool-2.0.7
ant initialize
```

3. Open **/home/mycloud/assemblytool-2.0.7/assemble.properties** in an editor.
4. Add the following lines to the end of the file:

```
engine.root = ../engine/
plugins = ../plugins
```

5. Save your changes.

If you have created additional customer accounts/users, repeat the procedure described in this section for each user.

6.1.9 Make a preliminary build

You can now build your project by entering:

```
ece-build
```

This command may take a long time to execute. It downloads all the components required to build an Escenic project, clones the **mycloud** project from your Git host, and builds it.

6.1.10 Add missing configuration layers

Before actually building your project, the **ece-build** tool downloads all the required components, and also creates various configuration files if they do not already exist.

These configuration files include a set of Escenic **configuration layers**, which it creates in the **/home/mycloud/assemblytool-2.0.7/** folder. (For information about Escenic configuration layers and what they are for, see http://docs.escenic.com/ece-server-admin-guide/5.7-snapshot/configuration_layers.html.)

Currently, **ece-build** does not create enough configuration layers. If you look in **/home/mycloud/assemblytool-2.0.7/** you will see that it has created a **conf** folder with the following content:

```
conf
  Nursery.properties
  layers
    addon
      Files.properties
      Layer.properties
    common
      Files.properties
      Layer.properties
    default
      Files.properties
      Layer.properties
    family
      Files.properties
      Layer.properties
    host
      Files.properties
      Layer.properties
```

You need to add four similar layers, called:

```
environment
vosa
instance
server
```

In detail, you must do the following:

1. Create the layer folders:

   ```
   cd /home/mycloud/assemblytool-2.0.7/conf/layers
   mkdir environment vosa instance server
   ```

2. Copy **Layer.properties** from one of the default layers to each of the new folders. For example:

   ```
   cp  addon/Layer.properties environment/
   cp  addon/Layer.properties vosa/
   cp  addon/Layer.properties instance/
   ```

```
cp  addon/Layer.properties server/
```

3. Create a **`Files.properties`** file in each of the new layer folders:

```
touch environment/Files.properties
touch vosa/Files.properties
touch instance/Files.properties
touch server/Files.properties
```

4. Edit the contents of these new **`Files.properties`** files as described in section 6.1.10.1.
5. Edit the master file **`/home/mycloud/assemblytool-2.0.7/conf/layers/conf/Nursery.properties`** as described in section 6.1.10.2.

If you have created additional customer accounts/users, repeat the procedure described in this section for each user.

6.1.10.1 Editing Files.properties

The basic content of **`Files.properties`** is:

```
$class=neo.nursery.FileSystemDepot
fileSystemRoot = path
```

where *path* is different in each layer:

Layer	Path
`environment`	**`/etc/escenic/engine/environment/${com.escenic.environment "unknown"}/`**
`vosa`	**`/etc/escenic/engine/vosa/`**
`instance`	**`/etc/escenic/engine/instance/${com.escenic.instance "default"}`**
`server`	**`/etc/escenic/engine/server/${escenic.server "default"}/`**

6.1.10.2 Editing Nursery.properties

You can completely replace the content of **`/home/mycloud/assemblytool-2.0.7/conf/Nursery.properties`** with the following:

```
$class=neo.nursery.Bootstrapper

# ECE :: default layer
layer.01 = /layers/default/Layer

# ECE :: plugins layer
layer.02 = /layers/addon/Layer

# VOSA :: common VOSA/SaaS layer
layer.03 = /layers/vosa/Layer

# Customer :: common layer
layer.04 = /layers/common/Layer

# Customer :: family/group layer (e.g. presentation, editorial)
layer.05 = /layers/family/Layer
```

```
# Customer :: environment layer (e.g. testing, staging, production)
layer.06 = /layers/environment/Layer

# Customer :: host specific layer
layer.07 = /layers/host/Layer

# Customer :: instance specific layer
layer.08 = /layers/instance/Layer

# Customer :: "escenic.server" specific layer
layer.09 = /layers/server/Layer
```

6.1.11 Rebuild the project

Now that you have corrected the layer specifications, you can build the project again by entering:

```
ece-build
```

This time, the build will take less time, since all the Content Engine components and dependencies have already been downloaded. When it has finished, you will see that an .EAR file has been created in **`/home/mycloud/releases`** folder.

The basic command above builds from the master branch of your Git repo. You can build from other branches by specifying a **`-b`** option and the name of the branch:

```
ece-build -b branch-name
```

If you have created additional customer accounts/users, repeat the procedure described in this section for each user.

6.1.12 Configure a web server

When **`ece-build`** has finished, it displays a URL from which the generated EAR file can be downloaded. For example:

```
[ece-build-80] BUILD SUCCESSFUL! @ Fri Aug 28 09:26:50 CEST 2015
[ece-build-80] You'll find the release here:
[ece-build-80] http://builder-ip-address/mycloud/releases/mycloud-branch-master-
revb52cc94-2015-08-28_0925.ear
```

If you try to download the EAR file, however, you will see that the URL doesn't work. That's because you still need to configure a web server to serve the builds created in **`/home/mycloud/releases`**.

To configure the web server:

1. Revert to the **`root`** user by entering:

   ```
   exit
   ```

2. Open **`/etc/nginx/sites-available/default`** in an editor, and replace all the content in the file with the following:

   ```
   server {
     listen 81 default;
     include /etc/nginx/default-site/*.conf;
   }
   ```

3. Create a new folder called **`default-site`** in **`/etc/nginx/`**:

```
mkdir /etc/nginx/default-site
```

4. Create a file called **`mycloud.conf`** in **`/etc/nginx/default-site`**, and add the following content:

```
location /mycloud/releases/ {
    alias /var/www/mycloud/releases/;
    expires modified +310s;
}
```

   If you have created multiple customer accounts, repeat this step for each account.

5. Restart the nginx web server:

```
service nginx restart
```

You should now be able to download the EAR file you built from the link returned by **`ece-build`**.

You can also download the latest build from any branch of any project using a URL like this:

```
http://builder-ip-address:81/customer/releases/customer-branch-latest.ear
```

6.2 Single-Project Builder

Creating a Single-Project Builder involves the following steps:

1. Log in to the **`mybuilder`** VM
2. Install various required software packages
3. Initialize the builder
4. Configure the builder
5. Enable Git access
6. Build the project

6.2.1 Log in to Your Builder VM

To log in to your Builder virtual machine:

1. Log in to the KVM host machine.
2. Use SSH to log in to the required guest VM. To do this you need to supply the path of the VM's SSH configuration file (**`ssh.conf`**) stored in **`/var/lib/vizrt/vosa/images/`***vm-name*. If your builder VM is called **`mybuilder`**, for example, then you should use the following command to log in:

```
sudo ssh -F /var/lib/vizrt/vosa/images/mybuilder/ssh.conf root@guest
```

6.2.2 Install required software packages

All of the software packages needed on the builder can be installed from **`apt`** repositories. It's a good idea, however, to install Maven manually, as the standard Maven **`apt`** package for Ubuntu contains a number of additional packages that are not needed.

So you need to:

1. Install the Oracle JDK using **apt-get**
2. Add the Escenic **apt** repository
3. Install all other **apt** packages

6.2.2.1 Install the Oracle JDK

Installing the Oracle JDK using **apt-get** is a slightly more complicated procedure than for most packages, because you have to agree to the Oracle license during the installation procedure. Enter the following commands:

```
apt-get install python-software-properties
add-apt-repository ppa:webupd8team/java
apt-get update
apt-get install oracle-java8-installer
```

The last command displays a license agreement that you need to agree to, using a character-based UI. Use your tab key to move the selection highlight between options, and press the spacebar to select them.

Verify that the JDK is installed and that you have the latest version by entering:

```
java -version
```

6.2.2.2 Add the Escenic apt repository

Add the Escenic SW repository as follows:

1. Add the Escenic repository's public key to your VM's **apt** sources keyring:
   ```
   curl --silent http://apt.escenic.com/repo.key | apt-key add -
   ```
2. Add the Escenic repository to the list of source repositories:
   ```
   echo "deb http://apt.escenic.com akita main non-free" | tee -a /etc/apt/
   sources.list
   ```
3. Update your local **apt** database with the changes:
   ```
   apt-get update
   ```

6.2.2.3 Install other apt packages

All the remaining **apt** packages can be installed with a single command:

```
apt-get install escenic-common-scripts escenic-build-scripts ant zip unzip subversion
 git-core xml2 alien fakeroot
```

6.2.3 Initialize the Builder

To initialize the Builder, enter the following command:

```
ece-local-builder initialize
```

This generates a configuration file in the **/etc/escenic** folder:

/etc/escenic/builder.conf

6.2.4 Configure the Builder

To configure the Builder:

1. Open **/etc/escenic/builder.conf** in an editor, and edit the following lines:

```
customer_name=customer-name
technet_user=technet-user
technet_password=technet-pass
maven_vizrt_user=maven-user
maven_vizrt_password=maven-pass
git_url=ssh://git@git-ip-address/mycloud.git
ear_base_url=http://builder-ip-address
```

   where:

2. *customer-name is t*he name of the customer account, **mycloud** in this case.
3. *technet-username* and *technet-password* are your credentials for accessing Escenic Technet.
4. *maven-username* and *maven-password* are your credentials for accessing the Escenic Maven repo.
5. *git-ip-address* is the IP address of your Git host
6. *builder-ip-address* is the IP address of your builder host (that is, the host you are currently working on).

If you don't have all the necessary credentials, contact Escenic Support.

6.2.5 Enable Git access

The **mycloud** user needs access to Git in order to be able to build your project. So you need to:

1. Create an SSH key pair for the user.
2. Add the SSH public key to the list of Git users using Gitolite on the Git VM and grant the user read access to the **mycloud** repo.

In detail, the procedure is as follows:

1. Enter the following command to create a key pair:

```
ssh-keygen -t rsa -b 4096 -C "your-email@example.com"
```

   ssh-keygen outputs three prompts: you should respond to all of them by pressing **Enter**:

```
Enter file in which to save the key (/home/mycloud/.ssh/id_rsa): [Press enter]
Enter passphrase (empty for no passphrase): [Press enter]
Enter same passphrase again:  [Press enter]
```

   It is particularly important that you do not specify a passphrase, as the builder needs to be able to access Git without any human intervention.

   After you have responded to these prompts, the keys are generated and **ssh-keygen** finishes by outputting a fingerprint of the public key, for example:

```
The key fingerprint is:
a2:53:50:1d:c7:7e:c6:00:e8:a2:ca:c0:cb:d9:cf:f1 your-email@example.com
```

   You do not need to use this fingerprint for anything.

2. List the content of **/home/mycloud/.ssh/id_rsa.pub** by entering

```
cat /home/mycloud/.ssh/id_rsa.pub
```

3. Select the listed content and copy it. Make sure you copy all of the content, including **ssh-rsa** at the beginning and the email address at the end.
4. Close the file again.
5. Back on your own PC, create an empty file called **mycloud.pub** in your **gitolite-admin/keydir** folder.
6. Paste the text you copied into the file and save it.
7. Open **conf/gitolite.conf** for editing.
8. Grant the **mycloud** user read-only access to the **mycloud** repo:

```
repo mycloud
     RW+ = your-user-name
     R   = mycloud
```

9. Commit and push the changes back to the Git host:

```
git add keydir/mycloud.pub
git add conf/gitolite.conf
git commit -m "Added mycloud build user"
git push -u origin master
```

6.2.6 Build the Project

To build the project, enter:

```
ece-build
```

This command may take a long time to execute. It downloads all the components required to build an Escenic project, clones the **mycloud** project from your Git host, and builds it.

When **ece-build** has finished, it displays a URL from which the generated EAR file can be downloaded. For example:

```
[ece-build-80] BUILD SUCCESSFUL! @ Fri Aug 28 09:26:50 CEST 2015
[ece-build-80] You'll find the release here:
[ece-build-80] http://builder-ip-address:81/mycloud/releases/mycloud-branch-master-
revb52cc94-2015-08-28_0925.ear
```

Pasting the URL into a browser or using it in a **wget** command should download the EAR file.

You can also download the latest build from any branch of any project using a URL like this:

```
http://builder-ip-address:81/customer/releases/customer-branch-latest.ear
```

The basic **ece-build** command given above builds from the master branch of your Git repo. You can build from other branches by specifying a **-b** option and the name of the branch:

```
ece-build -b branch-name
```

7 Install Escenic

To install Escenic on a KVM guest machine, all you need to do is log in to your KVM host machine and:

1. Make sure that your target VM has been created with the required post-install hook scripts.
2. Edit the **service design package** (`sdp.xml` file) that defines the Escenic system to be installed.
3. Create an SDP configuration file (`sdp.conf`) alongside the `sdp.xml` file.
4. Run a `vosa` command to install the defined system on the selected guest VM.
5. Verify that the system has been installed correctly.

7.1 Verifying the post-install hook scripts

You can only install Escenic software on a VM if it was created with the required post-install hook scripts as described in section 3.4.2.1. If for some reason that is not the case, then you can fix it now, by throwing the VM away and recreating it with the required scripts. This is a very simple procedure.

To check that the `mycloud` VM has been created with the required hooks:

1. Open `/etc/vizrt/vosa/available.d/mycloud/install.conf` in an editor.
2. Check that the following commands are all there and not commented out:

```
postinstall wait-for-ssh.sh
postinstall make-a-motd.sh
postinstall run-sdp-bootstrapper.sh
```

If they are all there, then you can just exit the file. If not, then:

1. Correct the file and save it.
2. Enter the following sequence of commands:

```
sudo vosa -i mycloud stop
sudo vosa -i mycloud destroy
sudo vosa -i mycloud create
```

or alternatively, enter them all on one line as follows:

```
sudo vosa -i mycloud stop destroy create
```

7.2 Editing the service design package

A **service design package** is an XML file called `sdp.xml` that defines exactly what Escenic components are to be installed on a particular VM. A service design package is required because an Escenic system has a lot of optional components called plug-ins, and users do not always want to install all of them. Also, in production environments an Escenic installation is typically spread across several hosts with specialized functions (editorial hosts, presentation hosts, database hosts and so on). In this kind of installation, each host can have a different set of components installed on it.

To get up and running for development purposes, however, it is sufficient to install an all-in-one Escenic system on one VM.

To install an all-in-one system on the **mycloud** VM you have created:

1. Open the supplied service design package **/etc/vizrt/vosa/available.d/mycloud/sdp.xml** for editing.
2. Edit the file as described below.
3. Save it.

The supplied **sdp.xml** file contains a lot of information, most of which you can leave unchanged. You do, however, need to add a number of values as described in the following sections.

7.2.1 Global variables

Near the top the file you will find a **global-variables** element like this:

```
<global-variables>
  <param name="name">name</param>
  <param name="timezone">timezone</param>
  <param name="location">location</param>
  <param name="shortname">shortname</param>
  <param name="realm">realm</param>
  <param name="domain">domain</param>
</global-variables>
```

Fill in the fields as follows:

1. *name*: A descriptive name for this system definition, for example:

   ```
   <param name="name">My Escenic Cloud</param>
   ```

2. *timezone*: Your time zone string, for example:

   <param name="timezone">Oslo/Europe</param>

3. *location*: An address or location description, for example:

   ```
   <param name="location">Head Office, Akerbrygge</param>
   ```

4. *shortname*: A short name for this system definition, for example:

   ```
   <param name="shortname">mycloud</param>
   ```

5. *realm*: a string identifying the purpose of this installation, usually one of **development**, **uat** (=user acceptance test) or **production**. For example:

   ```
   <param name="realm">development</param>
   ```

6. *domain*: A domain name for the host, for example:

   ```
   <param name="domain">mycloud.mycompany.com</param>
   ```

7.2.2 Authentication credentials

Look for this element (also near the top of the file):

```
<feature type="authentication" revision="1" id="technet-login">
  <param name="url">http://technet.escenic.com/</param>
  <param name="userid">username</param>
  <param name="password">password</param>
```

```
</feature>
```

Replace *username* and *password* with a username and password for accessing Escenic Technet. If you don't have credentials, contact Escenic Support.

7.2.3 System user passwords

Search for the string **`unix-user`** to find these two elements, that are used to set the passwords for the system's **`root`** user and the user **`escenic`** (owner of the Escenic installation).

```
<module type="unix-user" revision="1" id="root-user">
   <param name="username">root</param>
   <param name="password">password</param>
</module>

<module type="unix-user" revision="1" id="escenic-user">
   <param name="username">escenic</param>
   <param name="password">password</param>
   <param name="group">adm</param>
</module>
```

Replace *password* with your preferred passwords.

7.2.4 Escenic apt repository passphrase

Search for **`additional-packages`** to find this element that lists various SW packages required by Escenic.

```
<module type="additional-packages" revision="1" id="jdk8">
  <param name="deb-package">unixodbc</param>
  <param name="deb-package">defoma</param>
  <param name="deb-package">java-common</param>
  <param name="deb-package">xsltproc</param>
  <param name="deb-package-list">http://secrets.vizrtsaas.com/static/packages/java/8/
server.urilist</param>
  <param name="passphrase">passphrase</param>
</module>
```

Replace *passphrase* with the passphrase required to access Escenic's **`apt`** repository. If you don't have the passphrase, contact Escenic support.

7.2.5 Builder IP address

Search for **`escenic-ear`** to find these two elements:

```
<feature type="escenic-ear" revision="1">
  <param name="url">...</param>
</feature>

<feature type="escenic-conf" revision="1">
  <param name="url">.../param>
</feature>
```

Set the **`param`** elements as follows:

```
<feature type="escenic-ear" revision="1">
```

```
  <param name="url">http://builder-ip-address:81/customer/releases/customer-branch-
master-latest.ear</param>
</feature>

<feature type="escenic-conf" revision="1">
  <param name="url">http://http://builder-ip-address:81/customer/release/vosa-
conf-customer-1-customer-branch-master-latest.deb</param>
</feature>
```

where:

builder-ip-address is the IP address of your Production Builder host.

customer is the name of the Builder **customer account** (**mycloud** in this case). For an explanation of the term **customer account**, see section 6.1.4.

7.2.6 Escenic Maven repository credentials

Search for **maven-repository** to find the following elements:

```
<feature type="maven-repository" revision="1">
  <param name="url">http://maven.escenic.com</param>
  <param name="userid">maven-user</param>
  <param name="password">maven-password</param>
</feature>

<feature type="maven-repository" revision="1">
  <param name="url">http://repo.dev.escenic.com/content/groups/trunk</param>
</feature>

<feature type="maven-repository" revision="1">
  <param name="url">http://maven.escenic.com/unstable</param>
  <param name="userid">maven-user</param>
  <param name="password">maven-password</param>
</feature>
```

Replace *maven-user* and *maven-password* (in both places) with your credentials for accessing the Escenic Maven repository. If you don't have the necessary credentials, contact Escenic support.

7.2.7 Environment names

Search for **environment** to find the following elements:

```
<environment name="environment-name" id="environment-id">
  <cluster name="cluster-name">
    <machine name="machine-name">
    ...
    </machine>
  </cluster>
</environment>
```

Replace *machine-name* with the name of the guest VM in which you are installing Escenic (for example, **mycloud**) Replace *cluster-name* and *environment-name* with different names, for example **cloud** and **cloud-dev**. *environment-name* and *environment-id* can be the same. So, for example:

```
<environment name="cloud-dev" id="cloud-dev">
  <cluster name="cloud">
```

```
    <machine name="mycloud">
      ...
    </machine>
  </cluster>
</environment>
```

7.3 Create an SDP configuration file

Before you can install Escenic, you also need to create an SDP configuration file alongside your **`sdp.xml`** file. The procedure is as follows:

Create a text file called **`/etc/vizrt/vosa/available.d/mycloud/sdp.conf`**.

Open the file for editing and add the following content:

```
REALM=realm
ENVIRONMENT=environment
CLUSTER=cluster
MACHINE=machine
```

where:

1. *realm* matches the realm name specified in **`sdp.xml`**.
2. *environment* matches the environment name specified in **`sdp.xml`**.
3. *cluster* matches the cluster name specified in **`sdp.xml`**.
4. *machine* matches the machine name specified in **`sdp.xml`**.

7.4 Install Escenic

Once you have finished editing **`sdp.xml`** and saved your changes, enter the following command to create the Escenic installation you have defined:

```
sudo vosa -i mycloud install
```

The installation can take a long time and generates a lot of output messages.

If the installation fails, it should output a message indicating what the problem is. If the error is one you can easily fix (a typo in **`sdp.xml`**, for example) then you can just fix the problem and enter the following command to destroy the VM, create a new one and install your revised system on it:

```
sudo vosa -i mycloud stop destroy create install
```

7.5 Verify the installation

If the installation is successful, then the **`vosa`** command outputs a message like this on exit:

```
[ece-install-23] Setting correct permissions on ECE related directories ...
[ece-install-23] Data directory root, /var/lib/escenic,
[ece-install-23] has correct permissions, skipping sub directories.
```

```
[ece-install-24] The installation is now complete! It took 0d 0h 0m 24s
```

You can now check the basic status of the VM by entering:

```
sudo vosa -i mycloud status
```

This should result in a message something like this:

```
mycloud enabled installed alive 22957 42:08
```

This tells you that the basic VM is installed and functioning correctly, but it doesn't tell you anything about the Escenic installation.

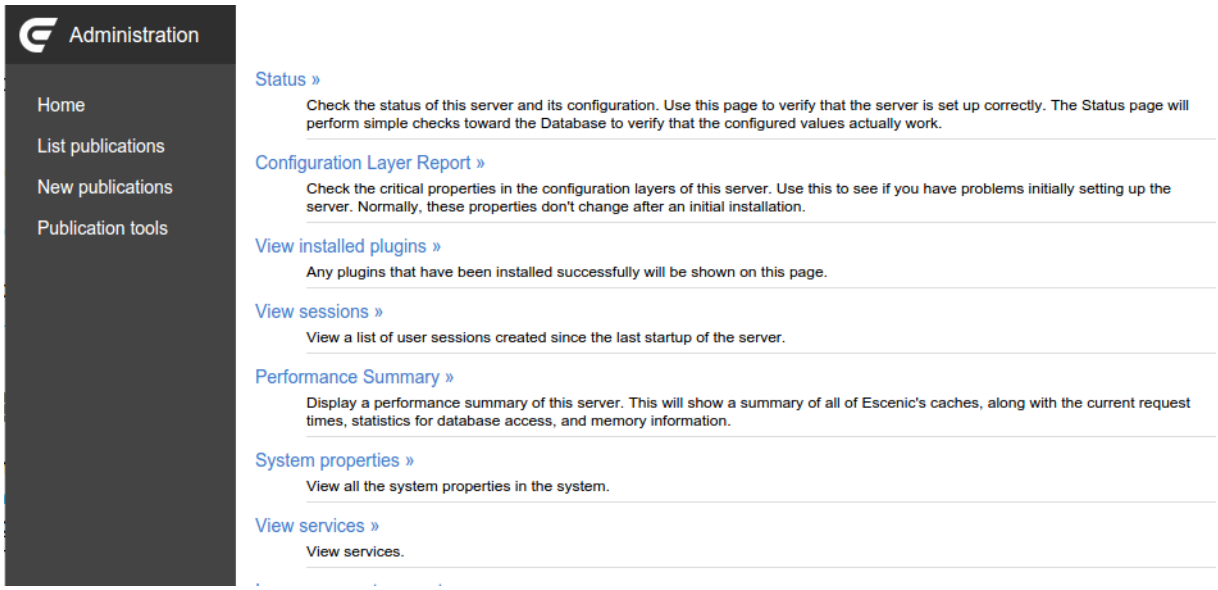
To verify the Escenic installation is working correctly:

1. Make a note of the following information:
   - The **`mycloud`** VM's IP address (defined in **`/etc/vizrt/vosa/available.d/mycloud/install.conf`**)
   - The Escenic installation's domain name (defined in **`/etc/vizrt/vosa/available.d/mycloud/sdp.xml`**)
2. Log in on your own PC.
3. Add the following entries to your hosts file:

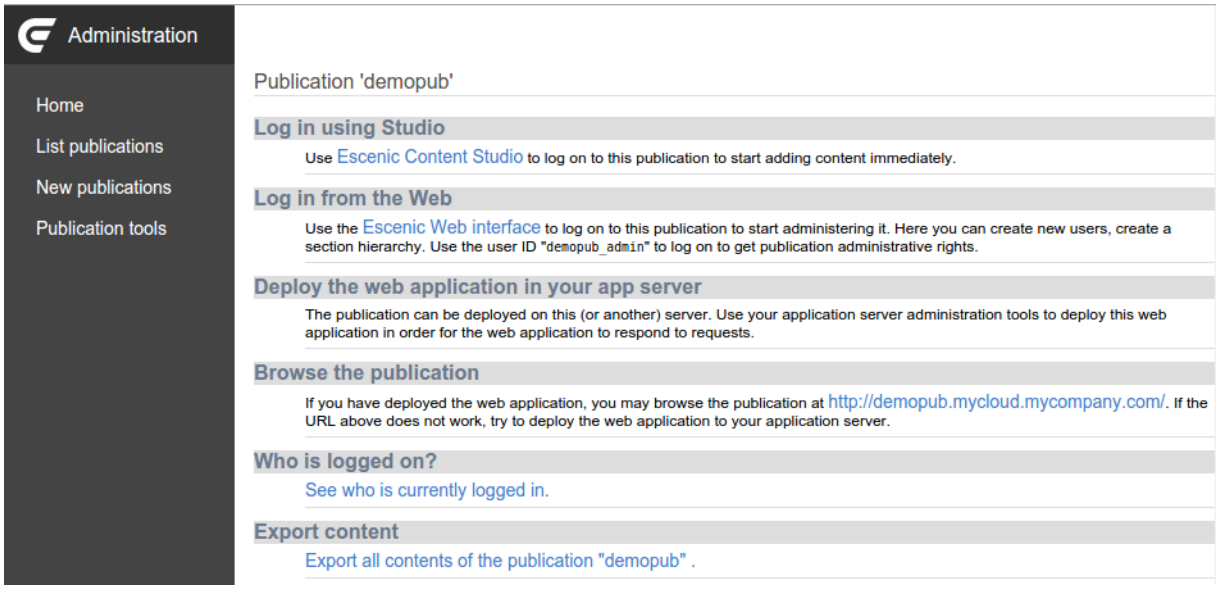
   ```
   ip-address:8080 domain-name demopub.domain-name
   ```

   replacing *ip-address* and *domain-name* with the values you noted down, for example:

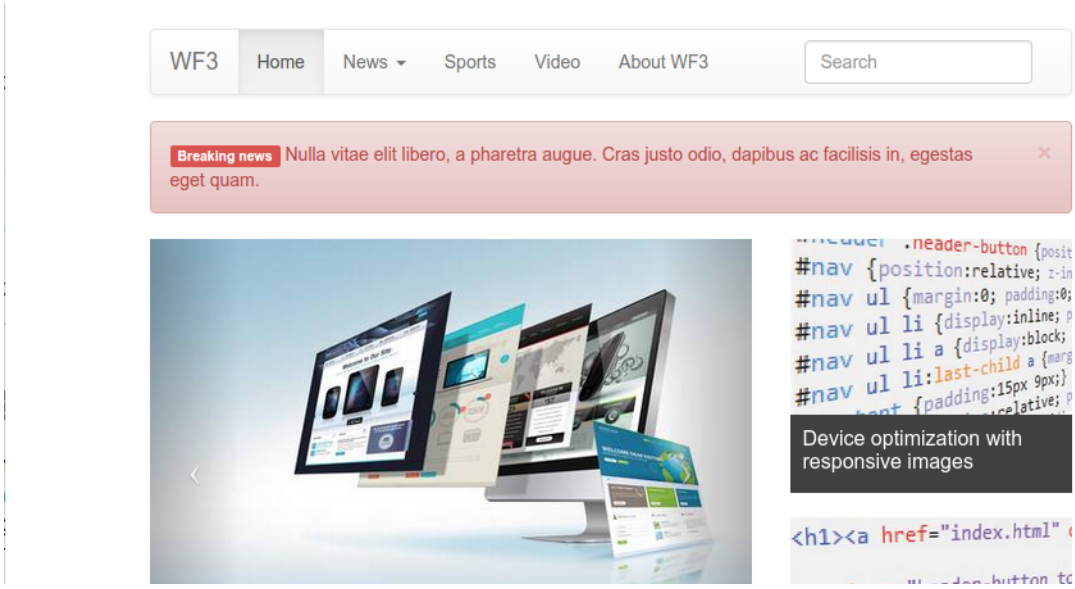
   ```
   10.136.68.45 mycloud.mycompany.com demopub.mycloud.mycompany.com
   ```
4. Open a browser and go to **`http://`***domain-name*`:8080/escenic-admin/`** (for example, **`http://mycloud.mycompany.com:8080/escenic-admin/`**). You should now see a standard **`escenic-admin`** start page looking something like this:



5. Click on **List publications** in the left-hand column; on the following page, click on the **demopub Information** link. You should then see a page like this:



6. Click on the link in the **Browse the publication** section of the page. This should display the Escenic Cloud **demopub** publication:



Congratulations! You now have a complete working Escenic Cloud installation. If you are going to use the installation for development purposes, however, then you will probably want to include a Development Builder in your **mycloud** VM as described in the following chapter.

8 Include a Development Builder

If you are using an Escenic installation for development purposes, then you are recommended to install a Development Builder in the same VM. Doing this allows developers to build locally and deploy changes directly to the Escenic system for testing instead of having to follow the full commit, push, build, deploy cycle. This local build process is much faster and simpler.

Including a Development Builder with the Escenic installation is very straightforward. All you have to do is:

1. Copy the private key of your Production Builder to your Escenic VM, as described in section 8.1.
2. Modify your **sdp.xml** as described in section 8.2.
3. If you've already installed Escenic, destroy and recreate the VM by entering:

   ```
   sudo vosa -i mycloud stop destroy create
   ```

4. Re-install from the modified **sdp.xml** file by entering:

   ```
   sudo vosa -i mycloud install
   ```

8.1 Copy the Production Builder key

To copy the Production Builder private key from the Production Builder VM (that is, **mybuilder**) to your Escenic VM (that is, **mycloud**), log in on your KVM host and enter the following commands

```
sudo scp -F /var/lib/vizrt/vosa/images/mybuilder/ssh.conf root@guest:/home/
mycloud/.ssh/id_rsa ./private-key
sudo scp -F /var/lib/vizrt/vosa/images/mycloud/ssh.conf ./private-key root@guest:/etc/
escenic/
rm ./private-key
```

This copies the private key to **/etc/escenic/private-key** on **mycloud**.

8.2 Modify sdp.xml

The additions you need to make to **sdp.xml** in order to install a Development Builder are already present in the file but commented out: all you need to do is to remove the comments around the additions described in the following sections and enter any missing details as specified.

8.2.1 Add an additional-package module

Search for the string **builder-pack** and remove the comments around the following block:

```
<module type="additional-packages" revision="1" id="builder-pack">
  <param name="deb-package">escenic-build-scripts</param>
  <param name="deb-package">subversion</param>
  <param name="deb-package">zip</param>
  <param name="deb-package">alien</param>
  <param name="deb-package">fakeroot</param>
</module>
```

This **module** element requests the installation of various Debian packages required by the Builder.

8.2.2 Add an escenic-builder module

Search for the string **escenic-builder** and remove the comments around the following block, which requests the installation and configuration of an Escenic Builder:

```
<module type="escenic-builder" revision="1">
  <use-feature refid="escenic-repo"/>
  <use-feature type="technet-login"/>
  <param name="customer">customer-name</param>
  <param name="builder-user">escenic</param>
  <param name="git-url">ssh://git-ip-address/mycloud.git</param>
  <param name="assemblytool-url">http://technet.escenic.com/downloads/release/57/
assemblytool-2.0.7.zip</param>
  <param name="ear-base-url">http://local-ip-address:81</param>
  <param name="machine">machine-name</param>
  <param name="branch">branch-name</param>
  <param name="maven-url">http://www.us.apache.org/dist/maven/maven-3/3.3.3/binaries/
apache-maven-3.3.3-bin.zip</param>
  <param name="private-key" src="/etc/escenic/private-key"/>
</module>
```

After uncommenting the element, make the following replacements:

1. Replace *customer-name* with the appropriate customer name (**mycloud**, for example).
2. Replace *git-ip-address* with the IP address of your Git VM.
3. Replace *local-ip-address* with the IP address of **this** VM (the VM on which you are installing Escenic and this Development Builder).
4. Replace *machine-name* with the name of the VM on which you want the built packages to be deployed. The single VM setup in the default **sdp.xml** file only contains one machine definition, so this is the name you must specify (**mycloud**, for example).
5. Replace *branch-name* with the name of the Git repository branch you want the Builder to build from (**master**, for example)
6. The Maven download URL supplied with the **maven-url** parameter (**http://www.us.apache.org/dist**) references an Apache backup mirror. You can probably improve the download speed by choosing a local mirror instead. For a list of official download mirrors, look here.

8.2.3 Add module references

Search for the string **refid="builder-pack"** and remove the comments around the following two elements:

```
<ref-module refid="builder-pack"/>
<ref-module type="escenic-builder"/>
```

These elements reference the two new modules you have added.

8.2.4 Add a technet-login feature

Search for the string **id="technet-login"** and remove the comments around the following two block:

```
<feature type="authentication" revision="1" id="technet-login">
  <param name="url"></param>
  <param name="userid">technet-username</param>
  <param name="userid">technet-password</param>
</feature>
```

After uncommenting the element, replace *technet-username* and *technet-password* with your Technet access credentials.

9 Create an Escenic development image

Once you have created an Escenic development installation (an Escenic installation that incorporates a Development Builder), it's very useful to be able to duplicate it for installation on developers' PCs. The **vosa** command includes an option that lets you do this. The **vosa** command has a **make** subcommand that can be used to make a distributable virtual machine image from an Escenic Cloud VM. This allows each developer in a team to run a personal copy of the development installation on their own PC using a VM hosting environment such as VMWare or VirtualBox.

To create a distributable image from an Escenic Cloud VM:

1. Log in on your KVM host.
2. Enter the following command to install the additional packages required by the **vosa make** command:

   ```
   sudo apt-get install virtualbox pv zerofree extlinux
   ```

3. Enter one of the following commands to create the distributable image:
   - ```
     sudo vosa -i host-name make
     ```

     where *host-name* is the name of VM you want to copy (**mycloud**, for example). This command generates a **.vmdk** file. VMDK is VMWare's native image format, although it is supported by other virtualization systems too.
   - ```
     sudo vosa -i host-name make ova
     ```

     This command generates both a **.vmdk** file and an **.ova** file. OVA (or OVF) is a open standard for packaging VM images, and is even more widely supported than VMDK.

   The generated files are output to the **/var/www/html/dev** folder.

   > The **vosa make** command automatically stops the VM before creating an image file, so you may need to start it again afterwards.

4. Distribute copies of the generated files as required.

10 Upgrading Escenic Cloud

To upgrade from an earlier version of Escenic Cloud to version 2.0-SNAPSHOT:

1. Log in on your KVM host.
2. Open the file **`/etc/apt/sources.list.d/escenic.list`** in an editor (you will need to use **`sudo`**).
3. Find the line containing the string:

   ```
   deb http://apt.vizrt.com/
   ```

   In full, this line will look like this:

   ```
   deb http://apt.vizrt.com/ codename main non-free
   ```

   where *codename* can be one of several code names from earlier Escenic Cloud releases.
4. Replace *codename* with **`akita`**:

   ```
   deb http://apt.vizrt.com/ akita  main non-free
   ```

Escenic Cloud is now upgraded to version 2.0-SNAPSHOT.