Secure Hybrid Cloud: Reference Architecture for OpenStack*

Intel conducted a proof of concept with Mirantis and IBM Cloud to create a reference hybrid cloud deployment to demonstrate that a hybrid cloud can support enterprise-level security.

**Introduction**

In 2014 Intel conducted a proof of concept (PoC) with Mirantis and IBM Cloud to create a reference hybrid cloud deployment—a combination of public off-premises and private on-premises hosting—to demonstrate that a hybrid cloud can support enterprise-level security. The three primary components of this cloud environment are Mirantis’ automated OpenStack* distribution, Intel® architecture-based servers that support Intel® Trusted Execution Technology (Intel® TXT), and IBM Cloud’s unique SoftLayer cloud infrastructure-as-a-service environment.

This document describes the steps necessary to create a similar deployment, which will help to simplify the task for those enterprises that want to take advantage of the agility and flexibility that hybrid cloud computing offers.

**Overview**

The hybrid PoC demonstrates the viability of building a secure OpenStack hybrid cloud using heterogeneous hardware managed through a single management platform.

**Configuration Summary**

Figure 1, on the following page, shows an overview of the solution architecture. Mirantis OpenStack is configured and automatically deployed by a Fuel* Master node under the control of Fuel’s web-based UI. The OpenStack node hardware, which consists of the Fuel Master node, Controller nodes, and Compute nodes, is connected using a Fuel Management logical network. The Fuel Master node provides DHCP (Dynamic Host Configuration Protocol) and PXE (Preboot eXecution Environment) boot services for this network, which conveys untagged traffic. OpenStack management, storage, and fixed (that is, virtual machine (VM)) networks may be physically separate or isolated with VLANs or tunnels.

Once OpenStack components are deployed, Fuel can be used to validate the new cloud. Administrators can then log on to the Fuel Master node to access OpenStack command-line configuration tools, which in this case are used to configure Intel TXT post-deployment. Once the on-premises and off-premises clouds are deployed, a single OpenStack Horizon* dashboard instance, running on the Controller node in the on-premises cloud, is used to manage both cloud regions.
As shown in the figure, the OpenStack Controller node in the on-premises cloud hosts OpenStack’s Keystone component, which provides authentication and end point identification services for both cloud regions and interoperates with the attestation server. (See the Appendix: OpenStack Project Code Names and Descriptions for more information on OpenStack component projects.)

Both the public off-premises cloud (hosted on IBM Cloud’s SoftLayer infrastructure) and private on-premises cloud deploy Mirantis OpenStack on server platforms with Trusted Platform Module (TPM) and the appropriate BIOS to utilize Intel TXT. The TPM along with the appropriate attestation creates a measured launch environment (MLE), in which VM workloads can be securely executed on the trusted compute nodes shown in Figure 1. Nova-Scheduler handles placing workloads on the appropriate VMs using information collected from the attestation server. To facilitate private management communication between Intel and SoftLayer, a VPN is established between the internal and external sites.

Planning the Deployment

We recommend adhering to a methodical, phased approach, in which one component is built and tested before building the next component. This document assumes that an enterprise has completed a planning stage before it starts the hybrid cloud project. During the planning stage, the following steps should be taken:

- **Set goals.** Discuss the business benefits of hybrid cloud computing with all stakeholders. Clearly delineate goals and set milestones and metrics for achieving these goals.

- **Identify hardware and software requirements.** For example, not all servers support hardware virtualization. Also, choices must be made about virtualization software, attestation SDK, and other supporting software. Examples of software required to support virtualization include KVM* (Kernel-based Virtual Machine),
a Linux* kernel module that allows an application to take advantage of Intel® Virtualization Technology;* QEMU* (Quick EMUlator), a generic and open source machine emulator and virtualizer; and Libvirt*, a virtualization API that interacts with the virtualization capabilities of the OS.

- **Verify that appropriate and adequate resources exist.** Resources to consider include staff to support the hybrid cloud, project funding, physical servers, data center space, and network bandwidth.

- **Plan networks and network scopes.** Because the hybrid cloud includes two networks—one at the public cloud host and one on-premises—we recommend minimizing the number of times network address translation (NAT) is required. Most networks are designed to work well when one network uses NAT and the other network is the Internet; unexpected complications can arise when both ends use NAT. Evaluate IP addresses to identify any potential conflicts.

Once the planning has been completed, enterprises can use the following sections to guide their deployment and configuration of the on-premises private cloud, off-premises public cloud, and networking aspects of an OpenStack hybrid cloud.

### Building an On-Premises Private OpenStack Cloud

The following sections describe the steps involved in building the on-premises private cloud component of the hybrid cloud solution.

#### Determine Hardware Requirements

In addition to needing the basic hardware for the cluster, the reference architecture requires a pair of servers to host the attestation server and Mirantis Fuel deployment system.

#### Set Up and Configure the Mirantis OpenStack Deployment Environment with Fuel

Developed by Mirantis, Fuel is an open source, template-driven deployment engine that forms part of the Mirantis OpenStack distribution. To deploy Mirantis OpenStack, download the integral .iso and boot it into a physical server or a VM to create a Mirantis OpenStack Deployment Environment with Fuel (MOSDEF), which includes a DHCP server and PXE-boot server.

Slave nodes (target servers for your OpenStack deployment, attached to the same network) are then set to obtain IP addresses, and PXE-boot from the MOSDEF, which installs a bootstrap OS and software to facilitate deployment. Once this process completes, cloud configuration and deployment can be carried out mostly from Fuel’s web interface. The MOSDEF can rapidly, automatically, and repeatedly deploy robust OpenStack configurations based on a range of tested reference architectures, drawn from actual use cases.

The following steps summarize a simplest-case procedure for installing the MOSDEF and preparing slave nodes for OpenStack deployment. Actual installations with multiple network cards, unique PXE requirements, and other constraints may also require preconfiguration and configuration of networking during MOSDEF creation.

---

**For more information on Mirantis OpenStack and MOSDEF (Fuel) deployment, see the Mirantis OpenStack User Guide.**
At the beginning of the boot process, the Mirantis OpenStack Deployment Environment with Fuel (MOSDEF), boot screen displays. In this case, the MOS .iso is booting into an Oracle VirtualBox* VM.

To install Mirantis OpenStack on bare-metal hardware
1. From the Mirantis website, download the Mirantis OpenStack image.
2. Do one of the following:
   • Burn the provided ISO to a writeable DVD.
   • Create a bootable USB stick.
3. Insert (or mount through the IPMI (Intelligent Platform Management Interface)) the media you created into the server that will be your MOSDEF, and power on the system, just as you would for any OS installation.
4. Do one of the following:
   • Set the boot order for the system with the installation media as the first device.
   • Set the hard drive as the first device, and then select the location of the media that contains the installation file to install the software.
5. If necessary, you can modify the boot settings. To configure the IP address, default gateway, and DNS server for the MOSDEF, press the Tab key to display the Grub command line, and then edit that line.
6. Allow the boot process to finish.

When the installation has finished, remove or eject the installation media from the Fuel Master Node. It is especially important to do this if you set the boot order so that the USB/DVD drive boots before the hard disk.

olumes 1.29
Fuel Install (Static IP)
Press [Tab] to edit options.
Automatic boot in 20 seconds...

Caution: If you unintentionally boot from the installation media again (after step 6), you may damage or delete the environment.
As Fuel starts, the screen and boot messages above display.

To prepare the bare-metal nodes to boot from the MOSDEF
1. Attach the bare-metal nodes to the same L2 network (broadcast domain) as the MOSDEF, and then configure them to automatically boot via the network (PXE boot). Change the PXE network parameters as necessary. For more information, see the “Changing PXE Network Parameters During Installation” section of the Mirantis OpenStack User Guide.
2. Power on all slave nodes that you are going to use for the OpenStack environment. The slave nodes will obtain IP addresses from the MOSDEF’s DHCP server and PXE-boot a bootstrap environment that will enable OpenStack deployment.
3. After all slave nodes have booted, return to the MOSDEF and log on using the administrator user name and password that display on the boot screen. After logging on, use the `passwd` command to change the password.
4. With a browser running on a separate system (or VM) also attached to the L2 network, use the URL displayed on the MOSDEF boot screen to launch the Fuel UI. If you did not modify the IP address on the Fuel Setup screens, use the default URL: `http://10.20.0.2:8000/`
5. To log on, use the admin user name and the Fuel password you set in the MOSDEF setup screens. If you did not set a Fuel password during installation, log on using the default username and password. You can use the Fuel UI to change the password. Or if the server on which the MOSDEF is installed has more than one NIC, you can use the second NIC to access the Fuel UI. To do this, connect the NIC to the appropriate switch, and then set the IP address for this NIC.
6. Fuel auto-discovers available nodes. You can now create a new Mirantis OpenStack environment, using the Fuel UI to do most of the configuration.
Intel® Trusted Execution Technology

Intel® Trusted Execution Technology (Intel® TXT), an extension to the Intel® Xeon® processor, is designed to harden platforms against attacks to the hypervisor and BIOS, malicious rootkit installations, and other firmware and software attacks. Intel TXT establishes a root of trust, a hardware-based security foundation that is used to verify the integrity of other system components, such as the hypervisor.

Intel TXT helps protect virtualized server environments through isolation and attestation. At startup, Intel TXT measures the hash value of the hypervisor and compares it with a known good value. If the measurements do not match, indicating that the hypervisor may have been compromised, a policy could be set to block the launch. This enables the cloud service provider—or the private cloud—to establish pools of compute resources with proven integrity of server infrastructure on which tenant virtual machines run.

Deploy OpenStack and Test the Cluster

The next step is to install OpenStack components on the bare-metal systems, and then configure and start them. One Fuel Master Node can deploy and manage multiple OpenStack environments, but you must create each environment separately. This process is documented in detail in the Mirantis OpenStack User Guide.

Here are the basic steps:

1. To log on to the Fuel UI on the MOSDEF, open a browser window to http://10.20.0.2:8000/. Use the admin username and the Fuel password you created earlier; the password is admin if you did not reset it during installation.

2. To launch the wizard that creates a new OpenStack environment, click the New OpenStack environment icon in the Fuel UI.

3. (Optional) Modify the Fuel password.

4. Select the following:
   - A name for the environment, and then select the OS (Icehouse* on Ubuntu* 12.04.4 (2014.1.1-5.1))
   - Deployment mode (multi-node HA or non HA)
   - Hypervisor (a bare-metal installation will typically use KVM)
   - Network service (Nova-network, Neutron with GRE (Generic Routing Encapsulation) segmentation or Neutron with VLAN segmentation)

5. Select your storage back-ends for Cinder and Glance.

6. Select Additional related projects.

7. Click Create. Your new environment’s deployment template appears as an icon on the Fuel UI’s Environments home page.

8. Click the icon that represents your environment. Unallocated nodes are enumerated in the upper-right corner of the screen.

9. To see a list of unallocated nodes and the roles available to them, click Add Nodes. Allocate nodes first to the Controller function (up to three nodes in a full HA deployment), then to Compute/Storage or separately to the Compute and Storage functions. The Add Nodes wizard prevents most errors, but make sure that the OpenStack components are deployed optimally on the intended nodes.

10. Configure the following (for full details, see the appropriate section in the Mirantis OpenStack User Guide):
    - Networking for the cluster
    - Interfaces on slave nodes
    - Storage and partitioning on the Compute and Storage nodes

11. To run automated predeployment tests of the proposed OpenStack cluster’s network configuration, click the Verify Networks button at the bottom of the Network tab. You must correct any exposed errors before continuing.

12. To deploy your OpenStack cluster, click Deploy Changes.

13. After cluster deployment has successfully completed, run the automated cluster Sanity and Functional tests available from the Health Checks tab.
Enable and Configure Intel® Trusted Execution Technology

In addition to an Intel® Xeon® processor-based server, Intel TXT requires a hardware TPM and a digitally signed code module. Not all servers ship with an integrated TPM; the authenticated code module (ACM) may or may not be present. The presence of a TPM and the method of detection is supplier-specific and can vary significantly. Systems based on the Intel® Xeon® processor X5675 use a file-based ACM, downloadable from the Intel TXT website. Systems based on the Intel® Xeon® processor E5-2680 have moved from a file-based ACM to a firmware-based SINIT ACM. Most system manufacturers include the SINIT ACM firmware in their BIOS or the firmware is integrated when a server is ordered preconfigured with a TPM installed.

If the TPM is not present, Intel TXT will not display as a valid BIOS/system configuration option. You may need to check with your hardware supplier for specific instructions on enabling TXT for your platform. For the Intel, Mirantis, and IBM Cloud PoC, we needed to update the BIOS and apply the firmware-based SINIT ACM.

After the TPM has been installed and the ACM has been applied, Intel TXT will display as a configuration option in BIOS. If Intel TXT does not display as an option, double-check that you have followed the instructions that your hardware supplier provided.

In some cases the TPM must be cleared before useful information is returned. This will require a couple reboots (therefore, we recommend that you to check for valid information before going through the TPM clear process). Clearing the TPM also disables it, so after clearing you will need to go back into BIOS and re-enable TPM.

The following sections describe how to install and validate trusted boot and a trusted compute agent. These sections assume that the Ubuntu 12.04 Linux distribution with grub2 and, at the time of testing, OpenStack Havana compute hosting services have already been configured.

Install Trusted Boot

Trusted boot allows only signed software to run on the device. tboot is an open source, pre-kernel/virtual machine manager (VMM) module that uses Intel TXT to perform a measured and verified launch of an OS kernel or VMM.

1. Add the prerequisite packages.
   aptitude -y install tboot
   apt-get install trousers trousers-dbg libcurl3-openssl-dev chkconfig tpm-tools
   apt-get install openjdk-6-jdk openjdk-6-jre
2. Change the grub boot priority for trusted boot.
   mv /etc/grub.d/10_linux /etc/grub.d/20_linux
   mv /etc/grub.d/20_linux_tboot /etc/grub.d/10_linux_tboot
3. Modify the following settings in /etc/default/grub.
   vi /etc/default/grub
   change GRUB_HIDDEN_TIMEOUT=5 (The default is 0.)
   change GRUB_HIDDEN_TIMEOUT_QUICK=false (The default is true.)
   change GRUB_TIMEOUT=5 (The default is 10.)
4. Update grub.
   update-grub
5. Restart the system.
Validate Trusted Boot
To verify that trusted boot is functional, check to see that PCR0 and PCR17 contain data. (For more information, see the sidebar TPM PCRs.)

```bash
cat /sys/class/misc/tpm0/device/pcrs
txt-stat | grep "TXT measured launch"
cat /sys/class/misc/tpm0/device/owned
```

At this point, the TPM should be unowned or the trust agent registration will fail. PCR values should not contain all 0s or Fs.

```
PCR-00: E3 B3 FE B5 1D A8 90 BF 61 A8 60 E5 D1 01 B8 89 BE 6E A3
PCR-01: 3A 3F 78 0F 11 A4 B4 99 69 FC AA 80 CD 6E 39 57 C3 3B 22 75
PCR-02: 3A 3F 78 0F 11 A4 B4 99 69 FC AA 80 CD 6E 39 57 C3 3B 22 75
PCR-03: 20 9D 8E 89 33 FE 74 84 A6 52 D8 4D 1E CA 57 AA 41 3A 7F F2
PCR-04: 5D 4F 89 3D E5 0A 93 9E 8E 05 DA AB D0 E9 5A B3 3A 9E 62 1C
PCR-05: 3A 3F 78 0F 11 A4 B4 99 69 FC AA 80 CD 6E 39 57 C3 3B 22 75
PCR-06: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-07: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-08: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-09: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-11: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-12: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-13: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-14: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-15: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-16: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-17: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-18: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-19: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-20: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-21: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-22: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-23: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

The screenshot shows an example of a system that has a TPM installed and enabled, but the "FFs" that display in PCR-17 through PCR-22 indicate that tboot is not functioning.

If tboot is not functioning, use the `txt-stat` command to check for any issues. `txt-stat` creates a mass dump of all trusted boot information. If there are only a few lines in the file, trusted measurement isn’t working. You can use several TPM commands to help troubleshoot problems (TPM supports about 24 commands).

Note: if necessary, you can get a list of available TPM commands by typing the following:
```
root/[pathname] tpm <TAB>
```

Commands that may be especially useful for troubleshooting include the following:
```
tpm_version
tpm_getpubek
tpm_selftest -r -l debug
```
Set Up Attestation

After TPM is enabled and configured, the next step is to install, configure, and validate the trusted compute agent, which provides attestation. The Intel PoC used Mt. Wilson, an Intel® product based on the open source OpenAttestation (OAT) project.

Attestation software is a middleware solution that enables data centers to attest to the trust of hypervisors and operating systems on multiple devices, including servers, clients, network, and storage components. The primary purpose of this attestation is for trust verification, remediation, reporting, and compliance in data center and cloud environments.

An attestation server reports trust levels, which the trusted filter and Nova-Scheduler use to determine whether a VM can be run on a host. There are three different trust levels:

- **Trusted.** The host is known by the attestation server and all values reported by the host are correct, and the system’s identity and trust have been established. A trusted host will be able to run VMs that require trusted hardware.
- **Untrusted.** The host was trusted at one point but is no longer trusted because of changes in its system. A VM will not launch on a system as long as it is untrusted.
- **Unknown.** The attestation server has no knowledge about the specific system. It is neither trusted nor untrusted. An unknown host can run any VM except for those that specifically require trusted hardware.

Figure 2 shows how the Nova-Scheduler interacts with the attestation server to place workloads on appropriate VMs.

**Key Solution Stack for Attestation Reference Implementation Components**

- Intel TXT-enabled host server(s)
- OpenStack Havana installation containing the virtualized compute infrastructure in the private cloud
- Attestation middleware (OAT) for the OpenStack hosts
- Ubuntu KVM server with connected Internet access
- SSH tool such as PuTTY or WinSCP

Figure 2. Nova-Scheduler uses trust level information from the attestation server to place workloads on the appropriate virtual machines.
**Deploy Attestation**

1. Create a new VM using the VMM.
2. Configure the white list.
3. Register each Compute node with the attestation server to have its PCR values compared against the white list(s).

**Enable the OpenStack Management Plane for Intel Trusted Execution Technology Administration**

Three areas of configuration are necessary for OpenStack to take advantage of the attestation server and Intel TXT:
- Nova-Scheduler
- (Optional) Horizon dashboard - requires modification of Horizon
- OpenStack flavors

We recommend that you validate these changes by launching two OpenStack instances: one that requires a trusted host and one that does not.

**Connect Nova-Scheduler with the Attestation Server**

Changes are necessary to allow the scheduler’s trust_filter to correctly communicate with the attestation server, because the default trust_filter uses a different communication syntax than the attestation server. It is also necessary to implement the logic that correlates trust requirements to trust levels.

1. Copy the attestation server’s certificate from the installer package to `/etc/nova/att_server.crt`
2. Configure the scheduler driver on the cloud controller(s).
   
   ```
   scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
   scheduler_available_filters=nova.scheduler.filters.all_filters
   scheduler_default_filters=RamFilter,ComputeFilter,TrustedFilter
   ```
3. Add a new section to `nova.conf` and point to the attestation server.
   ```
   [trusted_computing]
   attestation_server=10.160.58.154
   attestation_port=8181
   attestation_server_ca_file=/etc/nova/att_server.crt
   attestation_api_url=/AttestationService/resources
   ```
4. Modify/update the trusted scheduler as necessary, to address known issues.
5. Restart Nova-Scheduler.
   ```
   service nova-scheduler restart
   ```

**Configure the Horizon Dashboard**

In most cases, we have found it best to modify the Horizon UI in order to set the trust requirements of flavors while they are being created. In addition, some of the UI tables can be modified so that they display extra information such as what host an instance is running on and the trust requirement of the flavor that instance is running.

**Note:** For PoC purposes, each server can be evaluated separately, but use of a white list is preferable.

A **white list** is a gold or baseline set of PCR values gathered from a “known good” source, such as a server that has been placed on a trusted, isolated network and the supplier’s firmware updates that have all been verified through checksums.

**Note:** Making changes to the Horizon dashboard is optional; you can also use the command-line interface to define OpenStack flavors.
The Horizon UI enables you to create new OpenStack flavors.

Create OpenStack Flavors
An OpenStack flavor is an available hardware configuration for a server. Each flavor has a unique combination of disk space and memory capacity, information that OpenStack uses to determine on which host the VM will launch. You can create flavors using the Horizon UI or the command-line interface.

To use the command-line interface to create three new flavors with the extended attribute tag denoting a trusted, untrusted, or unknown host, follow these steps:

1. To display the existing flavor list for the chosen image, use the `nova flavor-list` command and select the flavor to duplicate and create as a trusted flavor.

   ```
   <user-name>@host-name07: ~$ nova flavor-list
   +-------------------------------------------------------------------------------------------------+
   | ID  | Name     | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
   +-------------------------------------------------------------------------------------------------+
   | 10  | Small-A  | 2048      | 0    | 0         |      | 1     | 1.0         | TRUE      |
   | 100 | Mega-B   | 16384     | 0    | 0         |      | 8     | 1.0         | TRUE      |
   | 20  | Small-B  | 4096      | 0    | 0         |      | 1     | 1.0         | TRUE      |
   | 30  | Medium-A | 4096      | 0    | 0         |      | 2     | 1.0         | TRUE      |
   | 40  | Medium-B | 8192      | 0    | 0         |      | 2     | 1.0         | TRUE      |
   | 444 | Elastic-A| 65536     | 0    | 0         |      | 8     | 1.0         | TRUE      |
   | 445 | Elastic-B| 131072    | 0    | 0         |      | 8     | 1.0         | TRUE      |
   | 50  | Large-A  | 4096      | 0    | 0         |      | 4     | 1.0         | TRUE      |
   | 60  | Large-B  | 8192      | 0    | 0         |      | 4     | 1.0         | TRUE      |
   | 70  | Large-C  | 8192      | 0    | 0         |      | 4     | 1.0         | TRUE      |
   | 90  | Mega-A   | 8192      | 0    | 0         |      | 8     | 1.0         | TRUE      |
   | 9000| DO-NOT-USE-PaaS-A | 4096 | 20   | 0         |      | 2     | 1.0         | TRUE      |
   | 9010| DO-NOT-USE-PaaS-B | 8192 | 20   | 0         |      | 4     | 1.0         | TRUE      |
   | 9020| DO-NOT-USE-BOSH-A | 8192 | 20   | 0         |      | 4     | 1.0         | TRUE      |
   | 9030| DO-NOT-USE-PaaS-C | 8192 | 20   | 0         |      | 4     | 1.0         | TRUE      |
   +-------------------------------------------------------------------------------------------------+
   
   Sample screenshot of existing flavors output for the `nova flavor-list` command.
   ```

2. Create the new flavor, matching the existing flavor’s parameters and giving the new flavor a new ID (in this case, 9910) and then set is-public to FALSE. (It is often necessary to reference the flavor by ID number.)
3. To display the new flavor’s attributes, use the `nova flavor-show` command.

   ```
   nova flavor-create --is-public false MeasuredBoot-Small-A 9910 2048 0 1
   nova flavor-key 9910 set ‘trusted:trusted_host’='trusted'
   nova flavor-show 9910
   ```

   ![Sample screenshot of output for the `nova flavor-show` command.](image)

4. Use the `nova flavor-access-add` command to add a project.

   ```
   nova flavor-access-add 9910 '342ffb74248e43c2bc49a78a59a4bc89'
   ```

   ![Sample screenshot of output for the `nova flavor-access-add` command.](image)

5. Use the `nova flavor-show` command to verify that the new flavor has the correct attributes, especially the setting in the `extra_specs` column.

   ![Sample screenshot of output for the `nova flavor-show` command.](image)

6. To create the other two flavors—untrusted and unknown—repeat steps 1-5.

   ```
   nova flavor-create --is-public false MeasuredBoot-Small-A 9910 2048 0 1
   nova flavor-key 9910 set ‘trusted:trusted_host’='untrusted'
   nova flavor-create --is-public false MeasuredBoot-Small-A 9710 2048 0 1
   nova flavor-key 9710 set ‘trusted:trusted_host’='unknown'
   ```

   ![Sample screenshot of output for the `nova flavor-show` command.](image)
The Instances screen displays the existing OpenStack instances and can be used to verify that a new instance launched.

Validate the Setup

1. To access the Instances screen, go to `<openstackIP>/horizon/project/instances/`
2. To create a new instance, click **Launch Instance**.
3. Select an image (any available OS), give the instance a name, and change the Flavor field to the trusted flavor you created earlier. Then click **Launch**.
4. Using the `nova-show` command, identify the host on which the instance has launched.

```
<user-name>@host-name07: ~$ nova flavor-show 9910

+------------------------------------------------------------------------------------------------------+
| Property                        | Value                                             |
+------------------------------------------------------------------------------------------------------+
| status                          | SHUTOFF                                           |
| updated                        | 2014-08-20T18:44:53z                              |
| OS-EXT-STS:task_state          | None                                              |
| OS-EXT-SRV-ATTR:host           | host-name07                                       |
| image                           | W508R2_ApplicationServer (4fe4f350-2cf1-463a-ba9a-039fa882025d) |
| hostId                          | 4b4797026b96b561a595ed49a8f1910f4430279b43e5b8e2ed5 |
| OS-EXT-SRV-ATTR:instance_name  | instance-000003d1                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname | prdlintf0ece01.xxx.intel.com                      |
| flavor                          | Small-A (10)                                     |
| id                              | 9e81239-d330-488a-a35c-727dd24b9cf                |
| security_groups                 | [{"name": "default"}]                            |
| user_id                         | 7070c2d2202404f4ba678155a4a6925                    |
| name                            | MyCloudOWC1                                       |
| created                         | 2013-08-16T16:31:04z                              |
| tenant_id                       | 258f4f8e68f9edc2e3e0d59a52164db                    |
| OS-DCF:diskConfig              | MANUAL                                            |
| metadata                        | {}                                                |
| accessIPv4                      |                                                    |
| accessIPv6                      |                                                    |
| External_network                | 10.67.25.253                                      |
| OS-EXT-STS:power_state         | 4                                                 |
| OS-EXT-AZ:availability_zone     | az1                                               |
| config_drive                    |                                                   |
+------------------------------------------------------------------------------------------------------+
```

Sample screenshot of output for the `nova-show` command.

5. Verify that the new instance has launched on a Trusted host, and then check the trust status of the host on which the instance launched, using the attestation server portal.

The result should look similar to this screenshot.

6. If compute nodes are available that do not have Intel TXT enabled, you can also test the Unknown flavor. Repeat steps 1–6 again, but this time launch an instance using the value of “Unknown” for the Flavor setting.

This screenshot shows an instance launched on a different host that the attestation server does not recognize.
**Bare-Metal Server Provisioning in the Off-Premises Public Cloud**

The cloud hosting provider handles most of the tasks involved in building the off-premises public cloud component of a hybrid cloud, so it is important to choose the right provider. Once a choice has been made, many of the steps, such as configuring the Fuel Controller, deploying OpenStack, and testing the cluster, are the same as for the on-premises private cloud. However, certain aspects of the computing environment must be specially configured to support attestation and single-plane management.

**Choose a Public Cloud Hosting Service**

For the off-premises public cloud, the Intel PoC team selected IBM Cloud’s SoftLayer infrastructure, which has a hardware base of Intel TXT. When choosing a cloud service provider for building a hybrid cloud, verify that the provider is willing to collaborate to make changes to cluster networking to accommodate the recommended networking plan.

For scalability, security, and convenience, the cloud service provider should also offer the following features:

- Public, private, and management network interfaces
- OpenStack management, shared between internal and external clusters through a VPN
- Local VM connectivity
- Storage
- Public network access

**Determine the Hardware Requirements**

In addition to the basic hardware for the cluster, the reference architecture requires a VM to host Fuel.

**Set Up and Configure the Fuel Controller**

The process for configuring the Fuel controller in the off-premises public cloud is the same as the process described previously for the on-premises private cloud.

**Deploy OpenStack and Test the Cluster**

The process for deploying OpenStack and testing the cluster in the off-premises public cloud is the same as the process described earlier for the on-premises private cloud.

**Integrate Attestation**

The cloud hosting provider should have already automatically configured Intel TXT, so it is only necessary to establish connectivity between the attestation server, hosted in the on-premises private cloud, and the trusted compute pool in the off-premises public cloud. This process may require solving some networking issues that result from conflicts between OpenStack and the cloud hosting provider’s configurations. For example, because Fuel uses PXE, the service provider must support broadcast traffic.

To support the connection between the attestation server in the on-premises private cloud and the trusted compute pool in the off-premises public cloud, a VLAN on one side must be mapped to a VLAN on the other. See “Building Connectivity for Single-Pane Management” for more details.

---

**Mirantis Products and Services**

**Mirantis OpenStack**, based on the latest stable release of OpenStack, is an OpenStack distribution that remains close-to-community while achieving stability and performance through hardening, largely automated deployment (via Fuel) in proven cloud configurations (including HA configurations), and by including tested versions of plug-ins, drivers, and other tools for broad compatibility. Mirantis OpenStack is backed by world-class Mirantis support with guaranteed service-level agreements, plus extended support agreements from select Mirantis partners such as Canonical (Ubuntu), Cisco, Oracle, and VMware. Bottom line: Mirantis OpenStack offers users the benefits of broadly supported, production-ready OpenStack, while keeping things open and avoiding lock-in.

**Mirantis OpenStack Express** is an on-demand private-cloud-as-a-service. OpenStack Express makes it easy to get the private OpenStack cloud you need with the performance, security and, control you want, while minimizing capital investment. Editions of OpenStack Express are available for enterprises (featuring geolocation, full hardware customization, a-la-carte engineering, and managed services), for teams, and for solo OpenStack developers, learners, and evaluators. And all editions are backed by Mirantis support.

**Mirantis Services** comprise a full portfolio of offerings for custom deployment and solutions engineering. Experience across scores of deployments ensures that Mirantis customers get OpenStack running quickly and that cloud projects deliver continuous return on investment. Services include VMware and AWS migration, design and integration of IT-as-a-service and self-service solutions, Continuous Integration and Continuous Deployment (CI/CD), and consulting in OpenStack operations and DevOps.

**Mirantis Training** has helped hundreds of organizations become more self-sufficient and effective in deploying, managing, and adding value to OpenStack. Mirantis offers vendor-agnostic training in a wide range of formats, from brief courses to comprehensive, multiday boot camps, and offers two levels of certification examination for OpenStack Administrators (MCA100 - Associate and MCA200 - Professional).
Building Connectivity for Single-Pane Management

Once installed and configured, the two clouds should appear as regions in the Horizon dashboard (located on the on-premises private cluster), so that they can be uniformly administered from the same management platform. To enable this, the network configuration must permit management traffic to and from Horizon (on the on-premises private side) to reach off-premises public components on the appropriate management-traffic VLAN.

While there are different ways to set up the hybrid cloud network, depending on the network landscape and business needs, the following two basic ingredients create end-to-end connectivity:

- **Public off-premises to private on-premises traffic.** Set up a secure, encrypted VPN IPSec tunnel, using SoftLayer’s IPSec service or a Vyatta gateway appliance (which can serve as a firewall, router and gateway all in one). This IPSec tunnel should use GRE.

- **Private on-premises to public off-premises traffic.** Configure access to the private subnets on the private VLANs where OpenStack is located.

In some cases, you may want to use a direct connection between the two sites, instead of using a public VPN tunnel (which uses the Internet). This approach enables access to private on-premises servers without routing traffic over the Internet. Direct connections, such as IBM Cloud’s SoftLayer Direct Link service, are often used in the following scenarios:

- **Cost-effective bandwidth.** With extremely large volumes of data, direct connections can offer cost savings, because there is no data metering.

- **Consistent low latency.** For applications that are sensitive to latency, or that cannot tolerate variations in latency, a direct connection can provide reliable, consistent throughput.

- **Security.** Because direct connections do not use the Internet, they may be the right choice for highly sensitive workloads.

Even when using a direction connection, we recommend that the network configuration also include an IPSec tunnel for an additional layer of security.

Additional Resources

Use the following links to explore Intel TXT, Mirantis OpenStack, and OAT in more detail.

**Intel Trusted Execution Technology and Trusted Computing Resources**
- Intel® Trusted Execution Technology white paper
- IBM Cloud* Delivers Greater Security with Intel® TXT on the SoftLayer* Platform
- Trusted Compute Pools with Intel® Trusted Execution Technology
- Trusted Computing Pools
- Trusted Cloud computing with Intel TXT: The challenge

**Mirantis and OpenStack Resources**
- Mirantis OpenStack Documentation
- OpenStack Documentation

**OpenAttestation Resources**
- OpenAttestation SDK overview
- Steps to configure OpenStack to work with OAT

**IBM Cloud and Network Resources**
- SoftLayer, an IBM Company blog: Your Direct Link into SoftLayer Cloud
- SoftLayer, an IBM Company blog: Enterprise Customers See Benefits of Direct Link with GRE Tunnels
**Appendix: OpenStack Project Code Names and Descriptions**

This document references several OpenStack projects such as Horizon; the following table defines all the current OpenStack projects to reference (Juno, release #10).²

<table>
<thead>
<tr>
<th>CODE NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nova</td>
<td>OpenStack Compute</td>
</tr>
<tr>
<td>Neutron</td>
<td>OpenStack Networking</td>
</tr>
<tr>
<td>Swift</td>
<td>OpenStack Object Storage</td>
</tr>
<tr>
<td>Cinder</td>
<td>OpenStack Block Storage</td>
</tr>
<tr>
<td>Keystone</td>
<td>OpenStack Identity</td>
</tr>
<tr>
<td>Glance</td>
<td>OpenStack Image Service</td>
</tr>
<tr>
<td>Horizon</td>
<td>OpenStack Dashboard</td>
</tr>
<tr>
<td>Ceilometer</td>
<td>OpenStack Telemetry</td>
</tr>
<tr>
<td>Heat</td>
<td>OpenStack Orchestration</td>
</tr>
<tr>
<td>Trove</td>
<td>OpenStack Database</td>
</tr>
</tbody>
</table>

New capabilities under development for the Juno release and beyond

| Ironic    | Bare Metal                |
| Marconi   | Queue Service             |
| Sahara    | Data Processing           |
For more information on Mirantis, visit www.mirantis.com

For more information on IBM Cloud’s SoftLayer, visit www.softlayer.com

For more information on Intel’s cloud computing initiatives, visit intel.com/Cloud

Solution Provided By:

1 See the Intel solution brief, “Improving OpenStack*Hybrid Cloud Security.”
2 No computer system can provide absolute security under all conditions. Intel® Trusted Execution Technology (Intel® TXT) requires a computer with Intel® Virtualization Technology, an Intel TXT-enabled processor, chipset, BIOS, Authenticated Code Modules, and an Intel TXT-compatible measured launched environment (MLE). Intel TXT also requires the system to contain a TPM v1.s. For more information, visit www.intel.com/technology/security.
3 Devices supporting Intel® Trusted Execution Technology can be found listed in the Intel Trusted Execution Technology Server Platforms Matrix. The Mirantis OpenStack Hardware Bill-of-Materials Calculator can assist in finding hardware for the reference architecture from a range of sources, at a range of price-points.
4 Intel® Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, and virtual machine monitor (VMM). Functionality, performance or other benefits will vary depending on hardware and software configurations. Software applications may not be compatible with all operating systems. Consult your PC manufacturer. For more information, visit www.intel.com/go/virtualization
5 The proof of concept conducted by Mirantis, Intel Corp., and SoftLayer described in the solution brief “Improving OpenStack* Hybrid Cloud Security” used Neutron with GRE segmentation.
6 Any cloud service provider that offers a similar set of features could be used.
7 www.openstack.org/software/roadmap

THE INFORMATION PROVIDED IN THIS PAPER IS INTENDED TO BE GENERAL IN NATURE AND IS NOT SPECIFIC GUIDANCE. RECOMMENDATIONS (INCLUDING POTENTIAL COST SAVINGS) ARE BASED UPON INTEL’S EXPERIENCE AND ARE ESTIMATES ONLY. INTEL DOES NOT GUARANTEE OR WARRANT OTHERS WILL OBTAIN SIMILAR RESULTS.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS AND SERVICES. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS AND SERVICES INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY.

Intel, the Intel logo, Look Inside, the Look Inside logo, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.