



Get up and Running with
Virtualization & OpenStack

Presented By: **S. Saeid Hosseini**



[SayidHosseini](#)



[SayidHosseini](#)

Index

Virtualization

- Types of Virtualization
 - Operating System-level Virtualization
 - Operating System-level Virtualization softwares
 - VirtualBox

Introduction to OpenStack

- Landscape & Releases
- Keystone
- Glance
- Neutron

Index

- Nova
- Cinder
- Horizon
- Ceilometer
- Gnocchi
- Aodh
- CloudKitty

OpenStack Deployment

- Deployment Tools
- Introduction to DevStack

Index

- DevStack on Local VM
- Installing DevStack
- Notes on DevStack
- Why not to use DevStack
- Introducing Ansible
- What is OpenStack-Ansible
- OSA vs. DevStack
- OSA-AIO Requirements
- OSA-AIO Deployment
- OSA-AIO Deployment Notes
- Interfacing your OSA-AIO

Virtualization

In computing, **virtualization** refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources.

Virtualization began in the 1960s, as a method of logically dividing the system resources provided by mainframe computers between different applications. Since then, the meaning of the term has broadened.

Types of Virtualization

- Execution Virtualization
 - Hardware Level
 - Operating System Level
 - Programming Language Level
- Network Virtualization
- Storage Virtualization
- Desktop Virtualization
- ...

Hardware-assisted Virtualization

This term refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation.

This technique was originally introduced in the IBM System/370.

At present, examples of hardware-assisted virtualization are the extensions to the x86-64 bit architecture introduced with Intel-VT and AMD-V.

OS-level Virtualization

It offers the opportunity to create different and separated execution environments for applications that are managed concurrently.

Differently from hardware virtualization, there is no virtual machine manager or hypervisor, and the virtualization is done within a single operating system, where the OS kernel allows for multiple isolated user space instances.

OS-level Virtualization

The kernel is also responsible for sharing the system resources among instances and for limiting the impact of instances on each other.

A user space instance in general contains a proper view of the file system, which is completely isolated, and separate IP addresses, software configurations, and access to devices.

OS-level Virtualization Softwares

- VMWare Workstation Pro/Player
- Microsoft Hyper-V
- Parallels Desktop
- VirtualBox
- KVM
- QEMU
- Xen
- ...

VirtualBox



VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2.

Presently, VirtualBox runs on Windows, Linux, Macintosh.

Introduction to OpenStack



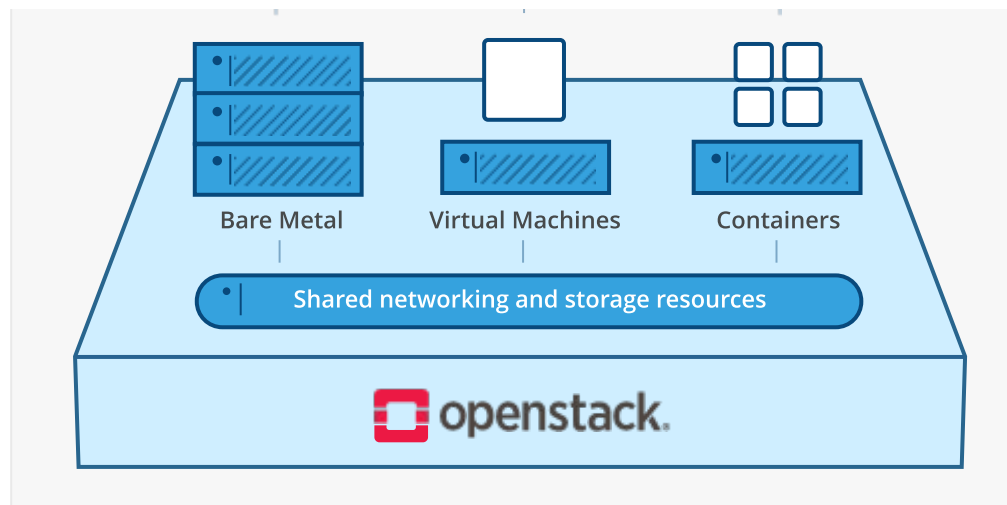
OpenStack is a **cloud operating system** that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common authentication mechanisms.

A dashboard is also available, giving administrators control while empowering their users to provision resources through a web interface.

Introduction to OpenStack



Beyond standard infrastructure-as-a-service functionality, additional components provide orchestration, fault management and service management amongst other services to ensure high availability of user applications.



OPENSTACK

WEB FRONTEND

Horizon

API PROXIES

EC2API

WORKLOAD PROVISIONING

Magnum Trove
Sahara

APPLICATION LIFECYCLE

Murano Freezer
Solum Masakari

ORCHESTRATION

Heat Mistral Aodh
Senlin Zaqr Blazar

COMPUTE

VIRTUAL MACHINES

Nova

CONTAINERS

Zun

FUNCTIONS

Qinling

NETWORKING

SDN
Neutron

LOAD BALANCING
Octavia

DNS
Designate

HARDWARE LIFECYCLE

BARE METAL
Ironic

ACCELERATORS
Cyborg

STORAGE

OBJECT
Swift

BLOCK
Cinder

FILE
Manila

SHARED SERVICES

Keystone

Placement

Glance

Barbican

Searchlight

Karbor

OPENSTACK-LIFECYCLEMANAGEMENT

DEPLOYMENT / LIFECYCLE TOOLS

Kolla-Ansible OpenStack-Charms TripleO Bifrost Kayobe
OpenStack-Helm OpenStack-Ansible OpenStack-Chef

PACKAGING RECIPES FOR...

RPM Puppet
Containers (LOCI, Kolla)

OPENSTACK-USER

SDK

OpenStackClient
Python SDK

OPENSTACK-ADJACENTENABLERS

CONTAINER SERVICES

Kuryr

NFV

Tacker

OPENSTACK-OPERATIONS

MONITORING TOOLS

Ceilometer
Monasca Panko

OPTIMIZATION / POLICY TOOLS

Watcher Vitrage
Congress Rally

BILLING / BUSINESS LOGIC

CloudKitty

MULTI-REGION TOOLS

Tricircle

openstack.

Bold represents **Core Functionality**

Version 2019.10.01

OpenStack Releases



Visit <https://releases.openstack.org> for the full list.

Here are the latest releases:

Series	Status	Initial Release Date	Next Phase
Victoria	Development	2020-10-14 <i>estimated (schedule)</i>	Maintained <i>estimated 2020-10-14</i>
Ussuri	Maintained	2020-05-13	Extended Maintenance <i>estimated 2021-11-12</i>
Train	Maintained	2019-10-16	Extended Maintenance <i>estimated 2021-04-16</i>
Stein	Maintained	2019-04-10	Extended Maintenance <i>estimated 2020-11-11</i>
Rocky	Extended Maintenance (see note below)	2018-08-30	Unmaintained <i>TBD</i>

Keystone



Keystone is an OpenStack service that provides API client **authentication, service discovery, and distributed multi-tenant authorization** by implementing OpenStack's Identity API.

Keystone is organized as a group of internal services exposed on one or many endpoints. Many of these services are used in a combined fashion by the frontend. For example, an authenticate call will validate user/project credentials with the Identity service and, upon success, create and return a token with the Token service.

Glance



The Image service (glance) project provides a service where users can **upload** and **discover data assets** that are meant to be used with other services. This currently includes images and metadata definitions.

Glance image services include **discovering**, **registering**, and **retrieving** virtual machine (VM) **images**. Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image.

Neutron



Neutron is an OpenStack project to provide “**network connectivity as a service**” between interface devices (e.g., vNICs) managed by other OpenStack services (e.g., nova). It implements the OpenStack Networking API.

Nova



Nova is the OpenStack project that provides a way to **provision compute instances** (aka virtual servers). Nova supports **creating virtual machines, baremetal servers** (through the use of ironic), and has limited support for **system containers**. Nova runs as a set of daemons on top of existing Linux servers to provide that service.

Cinder



Cinder is a **Block Storage** service for OpenStack. It's designed to **present storage resources to end users** that can be consumed by the OpenStack Compute Project (Nova). This is done through the use of either a reference implementation (LVM) or plugin drivers for other storage.

The short description of Cinder is that it **virtualizes the management of block storage devices** and provides end users with a self service API to request and consume those resources without requiring any knowledge of where their storage is actually deployed or on what type of device.

Horizon



Horizon is the canonical implementation of **OpenStack's Dashboard**, which provides a **web based user interface** to OpenStack services including Nova, Swift, Keystone, etc.

Horizon ships with three central dashboards, a "User Dashboard", a "System Dashboard", and a "Settings" dashboard. Between these three they cover the core OpenStack applications and deliver on Core Support.

Ceilometer



Ceilometer's goal is to efficiently **collect**, **normalize** and **transform** data produced by OpenStack services.

Ceilometer is a component of the **Telemetry** project. Its data can be used to provide **customer billing** (CloudKitty), **resource tracking** (Panko), and **alarming** capabilities (Aodh) across all OpenStack core components.

Gnocchi



Gnocchi is a **Metric-as-a-Service** project for OpenStack.

The problem that Gnocchi solves is the **storage and indexing of time series data and resources** at a **large** scale.

Gnocchi has been designed to **handle large amounts of aggregates** being stored **while being performant, scalable and fault-tolerant**.

Aodh



The Alarming service (Aodh) project provides a service that enables the ability to **trigger actions** based on **defined rules** against **metric or event data** collected by *Ceilometer* or *Gnocchi*.

Alarms can be triggered according to user-defined rules. This is especially **useful for autoscaling** services like *Heat*.

AODH is highly scalable horizontally.

CloudKitty



CloudKitty is a **Rating-as-a-Service** project for OpenStack and more. The project aims at being a generic solution for the chargeback and rating of a cloud.

CloudKitty allows to do **metric-based rating**: it polls endpoints in order to retrieve measures and metadata about specific metrics, **applies rating rules** to the collected data and **pushes the rated data** to its storage backend.

OpenStack Deployment



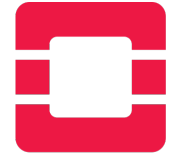
OpenStack consists of many underlying well-configured services.

To install (deploy) OpenStack *manually*, it would require a lot of expertise in the **Virtualization**, **Linux**, **Networking** and even **Security** areas.

Nowadays, it is **not** a good idea to deploy OpenStack *manually*, even for educational purposes.







There are a lot of automation tools that will take care of the installation for either **Development** or **Production** environments.

Deployment Tools





Heading to [OpenStack Deployment tools](#), you'll see:

Frameworks for lifecycle management

	TRIPLEO	Deploys OpenStack using OpenStack itself
	OPENSTACK-HELM	Deploys OpenStack in containers using Helm
	KOLLA-ANSIBLE	Deploys OpenStack in containers using Ansible
	KAYOBE	Deployment of containerised OpenStack to bare metal
	OPENSTACK-ANSIBLE	Ansible playbooks to deploy OpenStack
	OPENSTACK-CHARMS	Deploys OpenStack in containers using Charms and Juju
	BIFROST	Ansible playbooks using ironic
	OPENSTACK-CHEF	Chef cookbooks to build, operate and consume OpenStack

Packaging recipes for popular frameworks

	LOCI	Lightweight OCI containers
	PUPPET-OPENSTACK	Puppet modules to deploy OpenStack
	RPM-PACKAGING	RPM package specs to deploy OpenStack

Introducing the



DevStack is a series of extensible scripts used to quickly bring up a complete OpenStack environment based on the latest versions of everything from git master. It is used interactively as a development environment and as the basis for much of the OpenStack project's functional testing.

Check out the **Ussuri** release documentation from here:

<https://docs.openstack.org/devstack/ussuri/>

DevStack on a Local VM



- It's the time to create a virtual machine on your computer
- We are going to install Ubuntu 18.04 LTS on VirtualBox with the following minimum requirements:
 - 4 Virtual CPU Cores
 - 6 GB of RAM
 - 50 GB of Storage
- 2 Networks (NAT + Host-Only)
- Make sure to take a snapshot after a clean install
- Use SSH instead of console window, for easier communication

Installing



- Following the instruction from the DevStack documentation, first you'll need to create a non-root user with sudo privileges and change to that user:

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack  
$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack  
$ sudo su - stack
```

- If you can't access opendev, clone the GitHub repository instead:

```
$ git clone https://github.com/openstack/devstack.git -b stable/stein  
$ cd devstack
```

Installing



- Create a local.conf file with the following contents:

```
[[local|localrc]]
HOST_IP=w.x.y.z

ADMIN_PASSWORD=secret1
DATABASE_PASSWORD=secret2
RABBIT_PASSWORD=secret3
SERVICE_PASSWORD=secret4

LOGFILE=./stack.sh.log
VERBOSE=True
ENABLE_DEBUG_LOG_LEVEL=True
ENABLE_VERBOSE_LOG_LEVEL=True

GIT_BASE=${GIT_BASE:-https://github.com}
```

Installing



- Check the configuration file documentation from [here](#).
- The *HOST_IP* parameter defines the IP that horizon will be available on. Assign the IP from the *Host-Only* network card.
- Passwords can be the same or different.
- The logs of the installation will be available in *stack.sh.log* file in the devstack installation directory. Check that out if something goes wrong!
- The last line states that every repository should be downloaded from **GitHub**, should you not have access to opendev.

Notes on



- After preparing the file, enter *./stack.sh* to start the process. Make sure you are connected to the internet.
- If installation finished successfully, make sure to take a snapshot.
- If something went wrong, you always have the option to revert to your clean install snapshot.
- If you want to stop all of the devstack services, enter *./unstack.sh*.
- If you want to stop and uninstall devstack and all of its dependencies, enter *./clean.sh*.

Why not to use



- DevStack is a development environment with a relatively **huge gap** to a real production environment. It is better to choose an environment closest to the production.
- Many DevStack configuration regarding networking and storage **do not persist** after a reboot and you'll probably end up reinstalling it from scratch and losing your data.
- DevStack has a **monolithic inflexible** installer. If you need to add or remove a module or install it with a different configuration, you'll need to uninstall everything and then reinstall the whole thing.
- There is no OS level or even a package level isolation with DevStack.

Introducing Ansible



- Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.
- Ansible **models your IT infrastructure** by describing how all of your systems inter-relate, rather than just managing one system at a time.
- It uses **no agents** and **no additional custom security infrastructure**; all you need is *SSH* and *Python*.
- It uses a very simple language (YAML, in the form of **Ansible Playbooks**) that allow you to describe your automation jobs in a way that approaches **plain English**.

What is OpenStack-Ansible



- OpenStack-Ansible is an *official* OpenStack project.
- Its mission is to **deploy production environments** from **source** in a way that makes it **scalable** while also being simple to operate, upgrade, and grow.
- It provides Ansible playbooks and roles for the deployment and configuration of an OpenStack environment.
- It is possible to test OpenStack-Ansible on a single machine. It's called **OpenStack-Ansible All-In-One (OSA-AIO)** and it is a good replacement for DevStack.

OSA vs. DevStack



- OpenStack-Ansible deploys all the OpenStack services directly from their **git repositories**, without any vendor patches or add-ons. But the big difference is that OpenStack-Ansible deploys the OpenStack services in *LXC containers*, so there is **complete OS level and Python package isolation** between the services hosted on a node.
- Another difference between DevStack and OpenStack-Ansible is that the latter is a **production** distribution. With it, you can **deploy enterprise scale private clouds** that range from a handful nodes to large cluster with **hundreds or even thousands of nodes**.
- Of course it is immune to reboots. However, you may need to synchronize some services afterwards.

OSA-AIO Requirements



Absolute minimum server resources:

- 8 vCPU's
- 50GB free disk space on the root partition
- 8GB RAM

Recommended server resources:

- 8 CPU Cores with hardware-assisted virtualization capability
- 80GB free disk space on root partition / 60GB+ on a secondary disk
- 16GB RAM

OSA-AIO Deployment



- It is possible to perform AIO builds within a virtual machine *for demonstration and evaluation*, but your virtual machines will perform **poorly unless nested virtualization is available**.
- One disadvantage of deploying OSA-AIO instead of DevStack is that it **requires more resources**; that is caused by the underlying *lxc-container* infrastructure.
- Furthermore, not many PCs and laptops have the minimum requirements for the OSA-AIO deployments. So its best to deploy your OSA-AIO on an instance from a cloud provider.

OSA-AIO Deployment



- Install an Ubuntu Server 18.04 LTS and upgrade packages:

```
$ apt-get update  
$ apt-get dist-upgrade  
$ reboot
```

- **Clone** the OpenStack-Ansible repository and **checkout** to the branch you'd like to deploy on. Obviously, you do not want to deploy on the *master* branch since it might not be stable.

```
$ git clone https://opendev.org/openstack/openstack-ansible \  
    /opt/openstack-ansible  
$ cd /opt/openstack-ansible  
$ git checkout stable/ussuri
```


OSA-AIO Deployment



- Now with the *root* user, run the *bootstrap-ansible* script to bootstrap ansible and the required ansible roles:

```
$ scripts/bootstrap-ansible.sh
```

- After that, you'll need to prepare ansible with the appropriate disks partitioning, packages, network configuration and configurations for the OpenStack Deployment.
- By *default*, the AIO bootstrap scripts deploy a **base set of OpenStack services with sensible defaults** for the purpose of a gate check, development or testing system. For the default AIO scenario run:

```
$ scripts/bootstrap-aio.sh
```

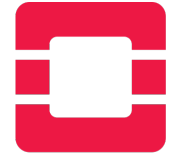
OSA-AIO Deployment



- To **add** OpenStack Services over and above the *bootstrap-aio* default services for the applicable scenario, **copy the *conf.d* files with the *.aio* file extension into */etc/openstack_deploy* and rename then to *.ym* files.**
- For example, in order to enable the OpenStack Telemetry services, execute the following:

```
$ cd /opt/openstack-ansible/  
$ cp etc/openstack_deploy/conf.d/{aodh,gnocchi,ceilometer}.yaml.aio \  
  /etc/openstack_deploy/conf.d/  
$ for f in $(ls -1 /etc/openstack_deploy/conf.d/*.aio); do mv -v ${f} ${f%.*}; done
```
- To add any **global overrides**, over and above the defaults for the applicable scenario, edit */etc/openstack_deploy/user_variables.yml*.

OSA-AIO Deployment



- Take a look at [this](#) document to learn how to implement **your own configuration** rather than to use the AIO bootstrap.
- Finally, run the **playbooks** by executing:

```
$ cd /opt/openstack-ansible/playbooks  
$ openstack-ansible setup-hosts.yml  
$ openstack-ansible setup-infrastructure.yml  
$ openstack-ansible setup-openstack.yml
```

- Once the playbooks have fully executed, it is possible to modify parameters in `/etc/openstack_deploy/user_variables.yml` and **only run individual playbooks**.
- For example, to run the *Keystone* service playbook, execute:

```
$ cd /opt/openstack-ansible/playbooks  
$ openstack-ansible os-keystone-install.yml
```

OSA-AIO Deployment Notes



- After a successful playbook run, you should see something like this:

```
PLAY RECAP *****
aio1 : ok=284 changed=127 unreachable=0 failed=0
aio1_cinder_api_container-37eccdc1 : ok=119 changed=57 unreachable=0 failed=0
aio1_glance_container-bd5530b2 : ok=96 changed=49 unreachable=0 failed=0
aio1_horizon_container-d232171c : ok=77 changed=44 unreachable=0 failed=0
aio1_keystone_container-7b8d2307 : ok=129 changed=63 unreachable=0 failed=0
aio1_neutron_server_container-fdaacd2b : ok=98 changed=55 unreachable=0 failed=0
aio1_nova_api_container-50b48737 : ok=113 changed=55 unreachable=0 failed=0
aio1_utility_container-76d83de7 : ok=84 changed=42 unreachable=0 failed=0
localhost : ok=3 changed=3 unreachable=0 failed=0

EXIT NOTICE [Playbook execution success] *****
=====
```

- The “*Playbook execution success*” is your key to go and certainly, you should not see any **unreachable** or **failed** numbers.

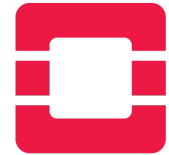
OSA-AIO Deployment Notes



- Since this is an AIO deployment and configurations are mostly pre-set, you should not encounter any errors if you follow the steps explained. However if you encountered an error and the log message was not enough, you could run the same playbook once more with **extended logs**. There are **four log levels** (this is an ansible feature) and by **default** you'll get **level-one** logs. To get more details, specify the **-vv**, **-vvv** or **-vvvv** at the end of your playbook execution command. For example to get the maximum log output for the first playbook, run:

```
$ openstack-ansible setup-hosts.yml -vvvv
```

OSA-AIO Deployment Notes



- The installation process will take a **while** to complete, and it is completely **dependent** on your **infrastructure** (*resources and internet bandwidth*), but here are some general estimates:
 - Bare metal systems with SSD storage: ~ 30-50 minutes
 - Virtual machines with SSD storage: ~ 45-60 minutes
 - Systems with traditional hard disks: ~ 90-120 minutes
- If you have **problem accessing some repositories** or you have limited connectivity issue, take a look at the “Installing with limited connectivity” document (*HTTP proxy* is a possible solution).
- Use either *tmux* or *screen* while running the playbooks. They will **keep your terminal session**, if you lost internet connectivity.

OSA-AIO Deployment Notes



- After playbooks execution, the SSH configurations will be modified and **you'll no longer be able to SSH via the root user using password**. Modify the SSH configuration (*/etc/ssh/sshd_config*) as you wish afterwards.
- As the AIO includes all three cluster members of MariaDB/Galera, **the cluster has to be re-initialized after the host is rebooted**. This is done by executing the following:

```
$ cd /opt/openstack-ansible/playbooks  
$ openstack-ansible -e galera_ignore_cluster_state=true galera-install.yml
```

- Should an existing AIO environment need to be reinstalled, the most efficient method is to destroy the host operating system and start over. For this reason, AIOs are best run inside of some form of virtual machine or cloud guest.

OSA-AIO Deployment Notes



- Sometimes it may be useful to **destroy** all the containers and **rebuild** the AIO. While it is **preferred** that the AIO is **entirely destroyed and rebuilt**, this isn't always practical. As such the following may be executed instead:

```
$ cd /opt/openstack-ansible/playbooks
$ openstack-ansible lxc-containers-destroy.yml
$ for i in \
    $(ls /etc/init | grep -e "nova\|swift\|neutron\|cinder" | awk -F'.' '{print $1}'); do \
    service $i stop; done

$ for i in $(pip freeze | grep -e "nova\|neutron\|keystone\|swift\|cinder"); do \
    pip uninstall -y $i; done

$ rm -rf /openstack /etc/{neutron,nova,swift,cinder} \
    /var/log/{neutron,nova,swift,cinder}
$ rm -rf /root/.pip
$ rm /etc/apt/apt.conf.d/00apt-cacher-proxy
```


Interfacing your OSA-AIO



- You can access your OpenStack through GUI or CLI.
- To access your OpenStack via GUI (**Horizon**), enter the *primary IP* address of the OpenStack machine into a browser, **accept the SSL error** and then you'll see the Horizon login page.
- Now on your OpenStack machine, run the following command to get the *admin password* and enter it on the login page:

```
$ cat /etc/openstack_deploy/user_secrets.yml | grep keystone_auth_admin_password
```

Interfacing your OSA-AIO



- OpenStack CLI client is fully installed on the *utility* container.
- To get a list of deployed containers, run:

```
$ lxc-ls -f
```

- To get inside the container, run:

```
$ lxc-attach -n aio1_utility_container-b289154f
```

- Here you'll see a file called *openrc* which includes the authentication parameters of admin user, required to work with the CLI interface. Before you can use the CLI you need to source the contents of this file with either of the commands:

```
$ . openrc  
Or  
$ source openrc
```

Resources

- Cloud-Computing course slides of Ferdowsi University of Mashhad
- <https://www.virtualbox.org/>
- <https://www.openstack.org/software/>
- <https://releases.openstack.org/>
- <https://docs.openstack.org/keystone/ussuri/>
- <https://docs.openstack.org/keystone/ussuri/getting-started/architecture.html>
- <https://docs.openstack.org/glance/ussuri/>
- <https://docs.openstack.org/neutron/ussuri/>
- <https://docs.openstack.org/nova/ussuri/>
- <https://wiki.openstack.org/wiki/Cinder>
- <https://wiki.openstack.org/wiki/Horizon>
- <https://wiki.openstack.org/wiki/Telemetry>
- <https://docs.openstack.org/ceilometer/ussuri/>
- <https://github.com/gnocchixyz/gnocchi>

Resources

- <https://docs.openstack.org/aodh/ussuri/>
- <https://www.noris.cloud/services/management-2/openstack-aodh/?lang=en>
- <https://docs.openstack.org/cloudkitty/ussuri/>
- <https://docs.openstack.org/devstack/ussuri/>
- <https://docs.openstack.org/devstack/ussuri/configuration.html>
- <https://stackoverflow.com/questions/20390267/installing-openstack-errors>
- <https://ask.openstack.org/en/question/58887/how-to-complete-uninstall-devstack/>
- <https://developer.rackspace.com/blog/life-without-devstack-openstack-development-with-osa/>
- <https://www.ansible.com/overview/how-ansible-works>
- <https://github.com/openstack/openstack-ansible>
- <https://docs.openstack.org/openstack-ansible/ussuri/>
- <https://docs.openstack.org/openstack-ansible/ussuri/user/aio/quickstart.html>
- <https://docs.openstack.org/openstack-ansible/ussuri/user/limited-connectivity/index.html>
- <https://docs.openstack.org/project-deploy-guide/openstack-ansible/ussuri/index.html>