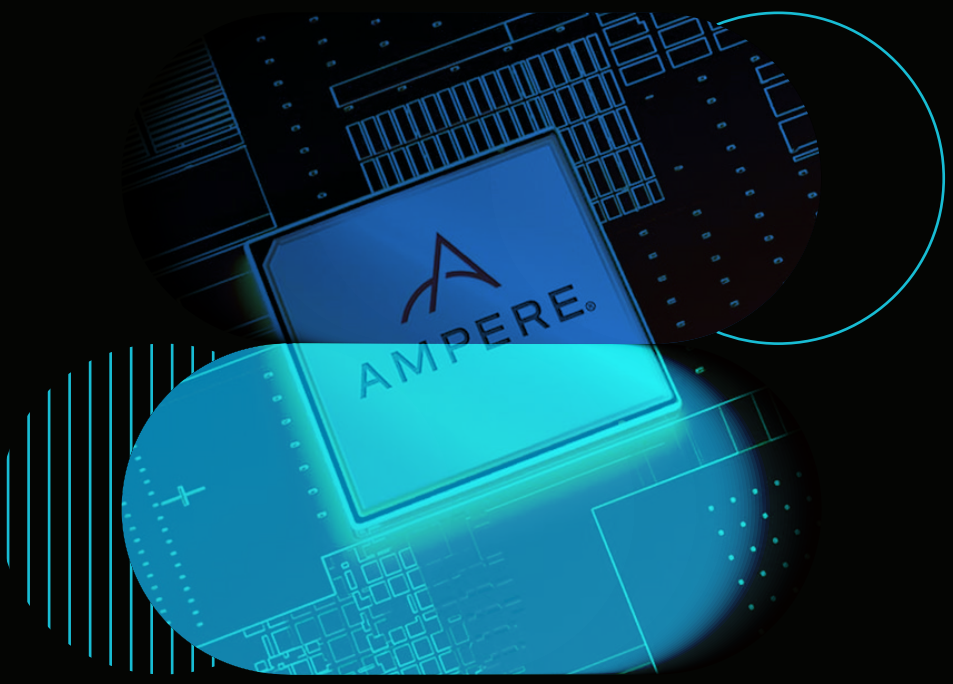
ShapeBlue and Ampere Reference Architecture for

IaaS Cloud on ARM64 Architecture



Introduction

As organizations increasingly build on-premises clouds or enter the Service Provider market with public clouds offerings, the need for the right set of tools to launch, manage and scale Infrastructure as a Service (IaaS) platform has never been greater. However, selecting the appropriate technology stack can be a complex process. From hardware to hypervisor, storage to cloud management platforms, the choices are vast and critical to ensuring a reliable and future-proof infrastructure.

When planning an IaaS platform, several factors must be considered, including future growth, team size, budget, project timelines, existing hardware, and the underlying infrastructure. Power consumption is also a crucial consideration, especially as the ARM64 architecture emerges as a compelling option.

The tech industry is seeing a trend towards ARM-based systems, especially with major companies like Apple switching to ARM for their Mac computers. This shift is helping to drive a broader market acceptance and pushing other companies to consider ARM as a viable alternative to x86.

Organizations are turning to Ampere® processors over the traditional x86 for several reasons, with a major consideration being the energy efficiency of its CPUs. Ampere processors are designed to be highly power-efficient, and the lower power consumption also means less heat generation, lower electricity bills for large infrastructures, and notable decreases in environmental impact. Ampere processors excel in performance per watt, making them ideal for environments where energy efficiency is paramount, such as data centers. The architecture allows for more cores per chip without significantly increasing power consumption.

ARM64 architecture is highly scalable, allowing cloud builders to deploy more cores per server without drastically increasing power or space requirements. This scalability enables higher server density in data centers, allowing providers to maximize the use of their physical infrastructure. Higher density means more computing power per square foot, which is essential for efficiently managing the vast workloads typical in cloud environments.

This Reference Architecture Guide offers a set of instructions to help you deploy a CloudStack-based IaaS cloud on ARM64 architecture powered by Mt. Collins by Ampere. The Mt. Collins provides a set of features that maximise performance and scalability to meet a diverse set of workloads. It allows to maximise application performance with an optimised mix of accelerator cards, storage and compute power in a 2U 2-socket server to meet diverse set of workloads.

Scope and Audience

Apache CloudStack is open-source software designed to deploy and manage large networks of virtual machines, as a highly available, highly scalable Infrastructure as a Service (IaaS) cloud computing platform. CloudStack is used by a number of service providers to offer public cloud services, and by many companies to provide an on-premises (private) cloud offering, or as part of a hybrid cloud solution.

CloudStack is a turnkey solution that includes the entire “stack” of features most organizations want with an IaaS cloud: compute orchestration, Network-as-a-Service, user and account management, a full and open native API, resource accounting, and a first-class User Interface (UI).

Ampere® Altra® and Altra Max® processors are designed to deliver high performance with single-threaded cores that run at consistently high frequencies and at large low-latency private caches. The architecture allows for high utilization and consistent performance even under heavy loads. These processors families were built from the ground up with an innovative scale-out architecture, featuring high core counts and efficient scaling. They are also highly power-efficient compared to x86 processors.

This document explores the process of building and IaaS cloud based on CloudStack and using Ampere Arm-based processors, highlighting the benefits of both for public and private cloud builders.

This document is designed for Cloud Architects, Solutions Architects, System Administrators, CTOs, CIOs, and companies that need to choose the right technology stack for building a high-performance cloud that is efficient, powerful, and reliable, all while managing infrastructure costs effectively.

Ampere and ShapeBlue

Ampere Computing and ShapeBlue have partnered to present an alternative approach to building IaaS with ARM and Apache CloudStack, instead of the traditional x86 servers. This collaboration focuses on introducing ARM-based infrastructure, which provides significant advantages in terms of total cost of ownership (TCO), efficiency, and performance. By utilizing ARM64 architecture, companies can benefit from lower energy consumption, reduced operational costs, and enhanced processing power, making it an ideal choice for modern cloud environments. This solution is tailored for organizations seeking to optimize their cloud infrastructure with cutting-edge technology.

With growing adoption and market momentum, ARM64 architecture is on track to become a choice for a wider range of cloud workloads. Although transitioning to ARM might present some challenges, the long-term benefits could be well worth the effort. As advancements in

ARM technology continue, its role in cloud computing is expected to expand, offering a powerful, efficient, and cost-effective alternative to traditional server architectures.

About Ampere Computing

Ampere Computing is a leader in delivering energy-efficient, high-performance processors designed for cloud-native environments. The Ampere Altra Max processors, featuring up to 128 high-performance cores per socket, provide exceptional processing power and scalability. These processors are optimised for modern data centre workloads, including data analytics, AI, and cloud services, with a strong focus on maximising performance per watt. This efficiency enables data centres to reduce operating costs and their overall carbon footprint, making Ampere an ideal choice for sustainable computing.

Ampere processors are engineered to meet the demands of today's cloud-based applications, with features such as eight channels of DDR4 memory support and 128 channels of PCIe Gen4 interfaces for rapid and reliable data transfer. Their architecture is specifically designed to support dense, scalable configurations, making them particularly well-suited for Infrastructure as a Service (IaaS) platforms.

About ShapeBlue

ShapeBlue is the largest independent integrator of CloudStack technologies globally and are specialists in designing and implementing IaaS cloud infrastructures for both private and public cloud implementations. The company combines 100s of person-years of experience in designing and building complex network, storage and compute infrastructures with globally leading skills in Apache CloudStack.

ShapeBlue offers services include consulting, integration, training and infrastructure support. Their technical teams are all active contributors to the Apache CloudStack project, and a good percentage of our team are also project committers.

ShapeBlue also work with and have deep expertise in a range of networking, storage and other related technologies that we have selected as ideally suited to the build-out of Clouds.

Document Overview

This reference guide offers a streamlined approach to deploying an Infrastructure as a Service (IaaS) platform using Ampere's ARM64 architecture. Leveraging Ampere's high-performance, energy-efficient processors, we have selected a configuration that utilises KVM on Ubuntu 22.04 LTS, NFS storage, and layer-2 isolation with VLANs. While CloudStack supports a broad range of technologies, this guide focuses on a simplified setup to ensure ease of deployment and management. It's important to note that production environments may require more complex configurations at a larger scale.

Concepts and Architecture

Overall CloudStack Concept

Apache CloudStack is an open-source cloud management platform (CMP) designed to deploy and manage extensive virtualized environments. It enables the creation of highly available, scalable, and resilient Infrastructure as a Service (IaaS) cloud. It supports a multi-architecture environment, compatible with both ARM64 and x86_64 processors, and offers broad compatibility with various hypervisors, such as KVM, Xen, and VMware. CloudStack's robust multi-tenant capabilities ensure secure resource isolation and efficient management of multiple users and organizations within the same cloud environment. With native self-service portals and automated resource provisioning, CloudStack is ideal for building private, public, edge, and hybrid cloud infrastructures.

Architecture Overview

CloudStack is a Cloud Orchestration platform that pools computing resources to build public, private and hybrid Infrastructure as a Service (IaaS) clouds. CloudStack manages the network, storage, and compute nodes that make up a cloud infrastructure.

A CloudStack cloud has a hierarchical structure which enables it to scale to manage tens of thousands of physical servers, all from a single management interface.

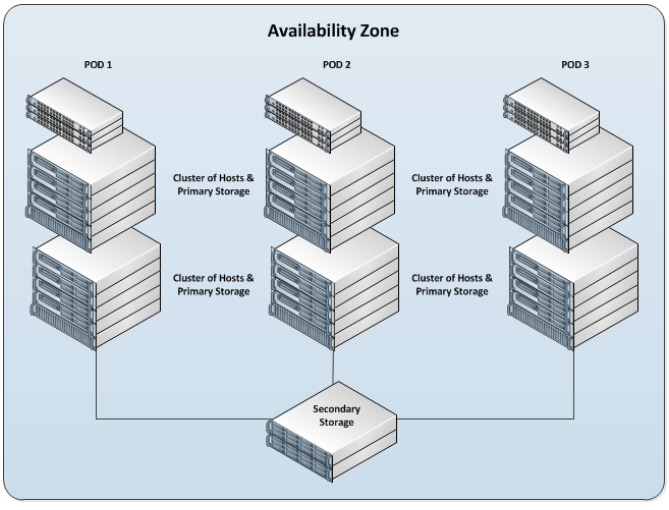


Figure 1: Example CloudStack Availability Zone

Zones

A CloudStack Zone (often called Availability Zone) is the largest organisational unit within a CloudStack deployment. Typically, a datacentre (DC) implementation will contain a single Zone, but there are no hard and fast rules, and a DC can contain multiple Zones. By

structuring CloudStack into geographical Zones, virtual instances and data storage can be placed in specific locations to comply with an organisation's data storage policies etc.

A Zone consists of at least one Pod, and Secondary Storage which is shared by all Pods in the Zone.

Zones are visible to end users, who can then choose which Zone they wish to create their virtual Instances in.

Pods

Typically, a Pod relates to a discrete rack in a datacentre so that from CloudStack a whole rack/pod of hosts can be taken offline for maintenance as a group. Pods contain one or more Clusters, and a Layer 2 switch architecture which is shared by all Clusters in that Pod. End users are not aware of and have no visibility of Pods.

Clusters

A Cluster is a group of identical Hosts running a common Hypervisor. For example, a Cluster could be a XenServer Pool, a group of KVM Servers or a VMware cluster pre-configured in vCenter. Each Cluster has a dedicated Primary Storage array which is where the virtual machine instances are hosted.

Primary Storage

Primary Storage is usually unique to each Cluster (although it could also be used Zone-wide) and is used to host Instances. CloudStack is designed to work with all standards-compliant iSCSI and NFS Servers supported by the underlying Hypervisor. Special storage solutions are also supported (such as CEPH, ScaleIO, SolidFire, etc). Primary Storage is a critical component and should be built on high-performance hardware with multiple high-speed disks.

Secondary Storage

Secondary Storage is used to store Instance Templates, ISO images and Volume Snapshots. The storage is available to all PODs in a Zone. Secondary Storage uses the Network File System (NFS) as this ensures it can be accessed by any Host in the Zone.

Instances

An Instance is a virtual machine that can be created by the end-user of CloudStack. Instances are based on Instance Templates and Service Offerings which specify the size (vCPU & RAM).

Installation and Setup

Overview

As far as the hardware goes, we got access to 2 x Ampere Mt. Collins systems. Each one has one 128-core Altra Max CPU @ 3.0Ghz, 256GB DDR4-3200, 960GB m.2 boot NVMe, 8x Samsung 30tb NVMe, 2x 100Gb network NICs. One of the servers will be used for the CloudStack management server and the related MySQL database server, while the other server will be used as a KVM hypervisor. Both servers will be running Ubuntu 22.04 as the base operating system. This will allow us to validate that CloudStack can be run purely on an arm64-based system and that there is no need to run any part of CloudStack on the standard x64 architecture.

As already stated, in this guide we will be using Ubuntu 22.04 LTS due to having KVM supported by default for the arm64 systems.

KVM, or Kernel-based Virtual Machine is a virtualization technology for the Linux kernel. KVM supports native virtualization atop processors with hardware virtualization extensions.

Setup Prerequisites

We will be using our Ampere servers in the following way:

- 1 x Mt. Collins (128 cores/256GB RAM/960GB boot drive + total of 240TB NVMe raw capacity).
    - On this server we will install CloudStack 4.19 management server + NFS server for exporting Primary and Secondary Storage Pools required for CloudStack
- 1 x Mt. Collins (128 cores/256GB RAM/960GB boot drive + some data drives, that won't be used in this setup)
    - This server will serve as KVM host, and will have CloudStack agent installed.
- We will need a bootable media for arm64-based Ubuntu 22.04 installation
- A /24 network with the gateway being at (e.g.) xxx.xxx.xxx.1, no DHCP is needed on this network and none of the computers running CloudStack will have a dynamic address. Again, this is done for the sake of simplicity. If you are using a different network range, that's no problem – the idea is that we will use a small part of this network for the POD/private IPs and a small portion of the same network to serve as a "Public" network in CloudStack. No need to turn off DHCP on this network, but we will set static IPs where needed.

Environment

Before you begin, you need to prepare the environment before you install CloudStack. We will go over the steps to prepare now.

Operating System

On both servers, install Ubuntu 22.04 LTS on your arm64 system. Ensure that the “universe” repository is enabled in /etc/apt/sources.list.

Install basic packages:

```
# apt update; apt install openntpd openssh-server sudo vim htop tar net-tools
```

Allow the “root” user for SSH access using password, by editing /etc/ssh/sshd_config and ensuring there is a line “PermitRootLogin yes”. Restart ssh daemon. Change and remember the root password, test remote access using SSH for the “root” user:

```
# passwd root
```

To disable issues down the road, make sure to stop and disable firewall:

```
# systemctl stop ufw; systemctl disable ufw
```

Configuring the network

Throughout this document, we are assuming that you will have a /24 network for your CloudStack implementation. This can be any RFC 1918 network. However, we are assuming that you will match the server address that we are using. Thus, we may use server IP 192.168.1.2 in this guide, and because you might be on an e.g. 172.16.10.0/24 network - you would use e.g. 172.16.10.2 or similar IP.

Connecting via the console or SSH, you should log in as root.

On the first server (we will call it Management server from now on),

- setup the static IP of 192.168.1.2 (in our example) – this can be done during the installation of the OS, or done manually later on

On the KVM host (the second server), we will start by creating the bridge that Cloudstack will use for networking.

First, let’s install the bridge utilities:

```
# apt install bridge-utils
```

Manually create a file at /etc/netplan/01-netcfg.yaml applying your network-specific changes:

```
network:
   version: 2
   renderer: networkd
   ethernets:
     eth0:
       dhcp4: false
       dhcp6: false
       optional: true
   bridges:
     cloudbr0:
       addresses: [192.168.1.3/24]
       gateway4: 192.168.1.1 #(this would be your physical/home router)
       nameservers:
         addresses: [8.8.8.8]
       interfaces: [eth0]
       dhcp4: false
       dhcp6: false
       parameters:
         stp: false
         forward-delay: 0
```

Note:

Interface name (**eth0**) is used as an example only. Replace “eth0” with your default ethernet interface name.
If your physical NIC has already been set up with static IP configuration before following this guide, make sure that there is no duplication between the IP configuration of cloudbr0 and eth0, which will cause a failure that would prevent the network from starting. Basically, IP configuration of eth0 should be moved to the bridge and eth0 will be added to the bridge (eth0 will have no IP configuration).

Save the file and apply network config, finally reboot:

```
# netplan generate
# netplan apply
# reboot
```

Hostname

CloudStack requires that the hostname is properly set. If you used the default options in the installation, then your hostname is currently set to localhost.localdomain. To test this, we will run:

```
# hostname --fqdn
```

At this point it will likely return:

```
localhost
```

To rectify this situation on both servers - we'll set a proper static hostname. We are using "mgmt.local" in our example and "kvm.local", for the management server and the KVM server, respectively.

```
hostnamectl set-hostname mgmt.local --static
```

```
hostnamectl set-hostname kvm.local --static
```

After you've modified that file restart both servers.

```
# reboot
```

Now recheck with the

```
# hostname --fqdn
```

and ensure that it returns a proper FQDN response.

Chrony

On both servers, turn on NTP for time synchronization.

Note: An NTP daemon is required to synchronize the clocks of the servers in your cloud.

Install chrony.

```
# apt install chrony
```

Configuring the CloudStack Package Repository

On both servers, we need to configure the machine to use a CloudStack package repository. We will be using CloudStack version 4.19.

On both servers

Import the gpg release key: (Key ID 584DF93F, Key fingerprint = 7203 0CA1 18C1 A275 68B1 37C4 BDF0 E176 584D F93F)

```
# apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
BDF0E176584DF93F
```

Add the following to your /etc/apt/sources.list.d/cloudstack.list file, to enable the repository:

```
deb http://packages.shapeblue.com/cloudstack/upstream/debian/4.19 /
```

Setup Storage

Our configuration is going to use NFS for both Primary and Secondary storage. We are going to set up two NFS shares for those purposes – both running from the management server (we can configure a RAID10 using 8 x 30 TB NVMe that came with this server and mount this under /export folder).

On the management server, install NFS server:

```
# apt install nfs-kernel-server quota
```

Create exports:

```
# echo "/export/primary *(rw,async,no_root_squash,no_subtree_check)" >> /etc/exports

# echo "/export/secondary *(rw,async,no_root_squash,no_subtree_check)" >> /etc/exports

# mkdir -p /export/primary /export/secondary

# exportfs -a
```

Configure and restart NFS server:

```
# sed -i -e 's/^RPCMOUNTDOPTS="--manage-gids"$/RPCMOUNTDOPTS="-p 892 --manage-gids"/g' /etc/default/nfs-kernel-server

# sed -i -e 's/^STATDOPTS=$/STATDOPTS="--port 662 --outgoing-port 2020"/g' /etc/default/nfs-common

# echo "NEED_STATD=yes" >> /etc/default/nfs-common

# sed -i -e 's/^RPCRQUOTADOPTS=$/RPCRQUOTADOPTS="-p 875"/g' /etc/default/quota

# service nfs-kernel-server restart
```

Management Server Installation

On the first/management server, we're going to install the CloudStack management server and the surrounding tools.

Database Installation and Configuration

We'll start with installing MySQL and configuring some options to ensure it runs well with CloudStack.

```
# apt update -y

# apt install mysql-server
```

Make a note of the MySQL server's root user password.

Configure InnoDB settings in mysql server's /etc/mysql/mysql.conf.d/mysqld.cnf:

```
[mysqld]
innodb_rollback_on_timeout=1
innodb_lock_wait_timeout=600
max_connections=350
log-bin=mysql-bin
binlog-format = 'ROW'
```

Restart MySQL server:

```
# systemctl restart mysql
```

CloudStack Management Server Installation

We are now going to install the management server. We do that by executing the following command:

```
# apt -y install cloudstack-management
```

CloudStack 4.19 requires Java 11 JRE. Installing the management server will automatically install Java 11, but it's good to explicitly confirm that Java 11 is the selected/active one (in case you had a previous Java version already installed):

```
# update-alternatives --config java
```

Make sure that Java 11 is the chosen one.

With the application itself installed we can now set up the database, we'll do that with the following command and options:

```
# cloudstack-setup-databases cloud:password@localhost --deploy-
as=root:<root password, default blank> -i <cloudbr0 IP here>
```

When this process is finished, you should see a message like "*CloudStack has successfully initialized database…*".

Important note:

As our IaaS platform is ARM64-based, we must seed an appropriate arm64 based SystemVM Template manually from the management server (instead of the default x64 SystemVM Template)

```
# wget
http://download.cloudstack.org/arm64/systemvmtemplate/4.19/systemvmtemplat
e-4.19.1-aarch64-kvm.qcow2

# /usr/share/cloudstack-common/scripts/storage/secondary/cloud-install-
sys-tmplt -m /export/secondary -f systemvmtemplate-4.19.1-aarch64-
kvm.qcow2 -h kvm -o localhost -r cloud -d password
```

Note:

When upgrading a ARM64 based CloudStack version, please ensure to keep the cloudstack-management stopped post-install/upgrade and manually copy the version-appropriate arm64-based SystemVM Template at */usr/share/cloudstack-management/templates/systemvm* folder and update its md5 checksum in the *metadata.ini*, on the management server host. Or just manually replace the x64 SystemVM Template qcow2 with the arm64 one in the secondary storage directory (under */export/secondary/template/tmpl/1/<new folder that contains the SystemVM Template>*).

Now that the database has been created and the arm64-based SystemVM Template is seeded, we can take the final step in setting up the management server by issuing the following command:

```
# cloudstack-setup-management
```

That concludes our setup of the management server. We still need to configure CloudStack, but we will do that after we get our hypervisor set up.

KVM Setup and Installation

Prerequisites

As already mentioned, we are going to use the second Ampere server as the KVM host. We have already performed most of the prerequisite steps previously, but we will list them here for clarity. Those steps are:

- Configuring the network
- Hostname
- Chrony
- Configuring the CloudStack Package Repository

(You don't need to do that now as we've already done that).

Installation

Installation of the KVM agent is trivial with just a single command, but afterwards, we'll need to configure a few things.

Install KVM and CloudStack agent, configure libvirt:

```
# apt install qemu-kvm cloudstack-agent cpu-checker
```

We have two different parts of KVM to configure - libvirt and QEMU

Enable VNC for console proxy:

```
# sed -i -e 's/\#vnc_listen.*$/vnc_listen = "0.0.0.0"/g'
/etc/libvirt/qemu.conf
```

CloudStack uses libvirt for managing virtual machines. Therefore, libvirt must be configured correctly. Libvirt is a dependency of cloud-agent and should already be installed.

Even though we are using a single host, the following steps are recommended to get familiar with the general requirements. To have live migration working libvirt has to listen for unsecured TCP connections. We also need to turn off libvirt's attempt to use Multicast DNS advertising

Configure default libvirtd config:

```
# echo 'listen_tls=0' >> /etc/libvirt/libvirtd.conf
# echo 'listen_tcp=1' >> /etc/libvirt/libvirtd.conf
# echo 'tcp_port = "16509"' >> /etc/libvirt/libvirtd.conf
# echo 'mdns_adv = 0' >> /etc/libvirt/libvirtd.conf
# echo 'auth_tcp = "none"' >> /etc/libvirt/libvirtd.conf
# systemctl restart libvirtd
```

We'll also need to disable apparmour on libvirtd:

```
# ln -s /etc/apparmor.d/usr.sbin.libvirtd /etc/apparmor.d/disable/
# ln -s /etc/apparmor.d/usr.lib.libvirt.virt-aa-helper
/etc/apparmor.d/disable/
# apparmor_parser -R /etc/apparmor.d/usr.sbin.libvirtd
# apparmor_parser -R /etc/apparmor.d/usr.lib.libvirt.virt-aa-helper
```

KVM configuration complete

For the sake of completeness, you should check if KVM is running OK on your machine.

Ensure that KVM is available at /dev/kvm:

```
# kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Important:

For the some arm64 CPUs (Ampere CPUs included), the CPU speed is not exposed via /proc/cpuinfo facility of the Ubuntu, so we have to manually specify the CPU frequency and

architecture via agent.properties file on the KVM side, in order for CloudStack to be aware of the CPU frequency/CPU capacity of the KVM host.

Please add the following lines to the */etc/cloudstack/agent/agent.properties* (you may need to check/re-add these for your KVM host after the zone deployment):

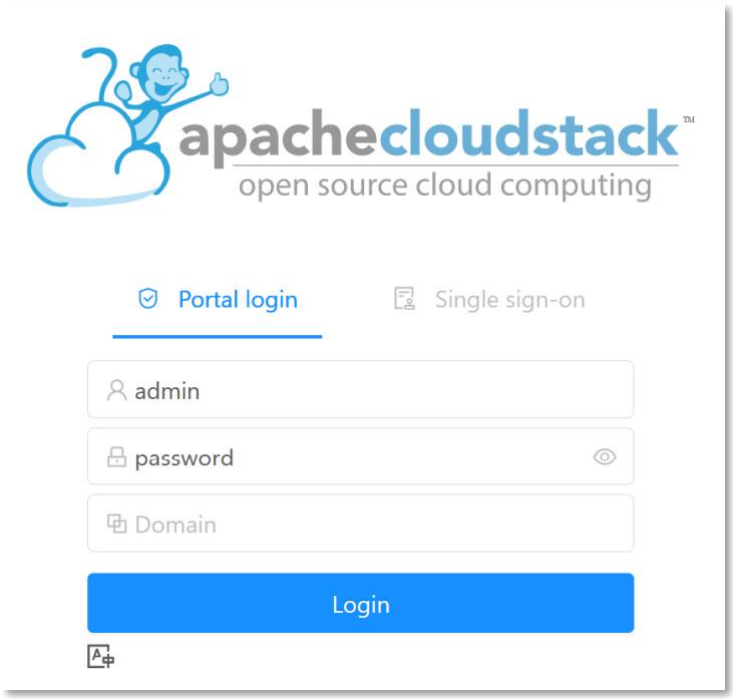
```
guest.cpu.arch=aarch64
guest.cpu.mode=host-passthrough
host.cpu.manual.speed.mhz=3000
```

That concludes our installation and configuration of KVM, and we'll now move to using the CloudStack UI for the actual configuration of our cloud.

Configuration of UI Access

To get access to CloudStack's web interface, point your browser to the IP address of your management server e.g. http://192.168.1.2:8080/client

The default username is 'admin', and the default password is 'password'



Setting Up a Zone

The following is an example of how you can setup an advanced zone in your local network - in our case, it's 192.168.1.0/24 network.

Setup a zone

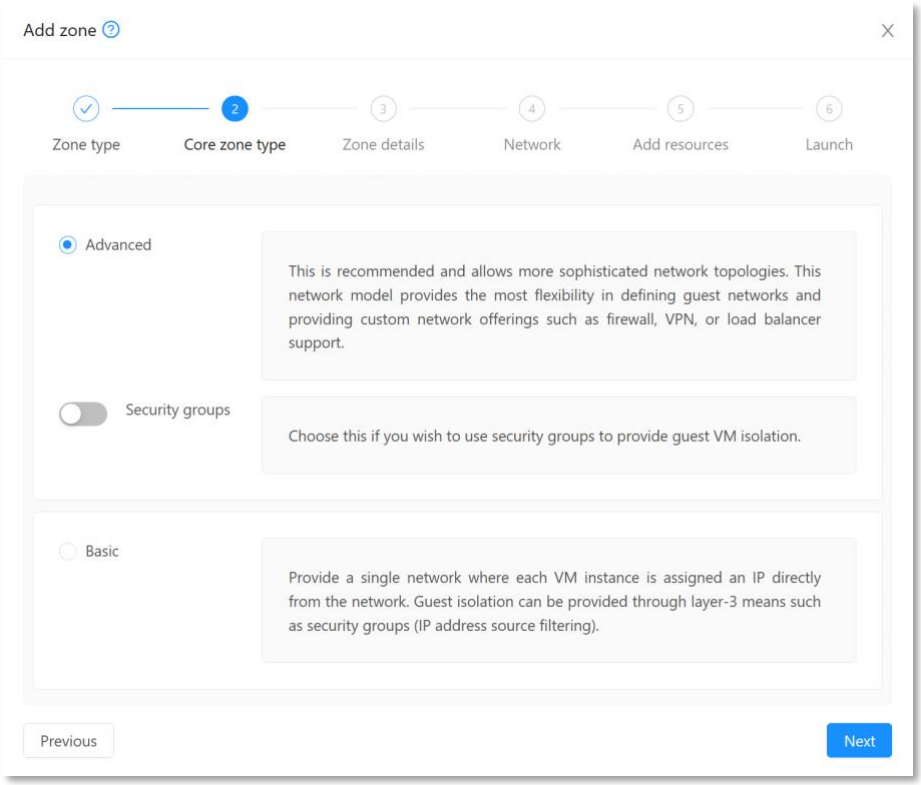
Go to Infrastructure > Zone and click on add zone button, select "Core", click on 'Next', select Advanced zone and provide the following configuration (leaving all other defaults):

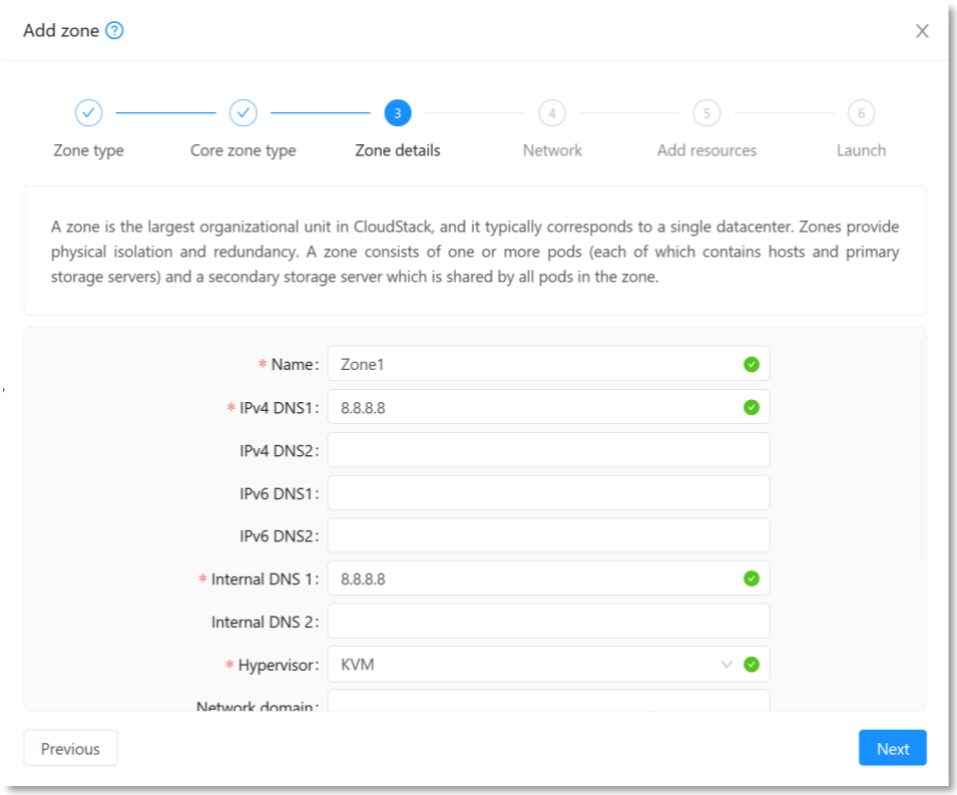
Name – Zone1

IPv4 DNS 1 - 8.8.8.8

Internal DNS 1 – 8.8.8.8

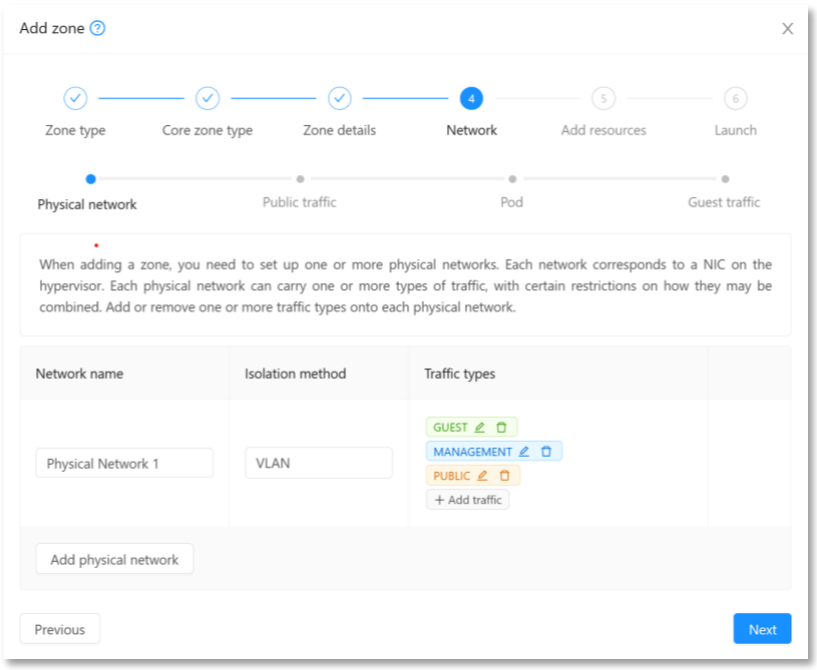
Hypervisor – KVM





Setup a network

On the next screen (Physical network details), use the defaults – we will use the VLAN isolation method on a single physical NIC (on the host) that will carry all traffic types (management, public, guest) and click “Next”.



Public traffic configuration:

Gateway - 192.168.1.1

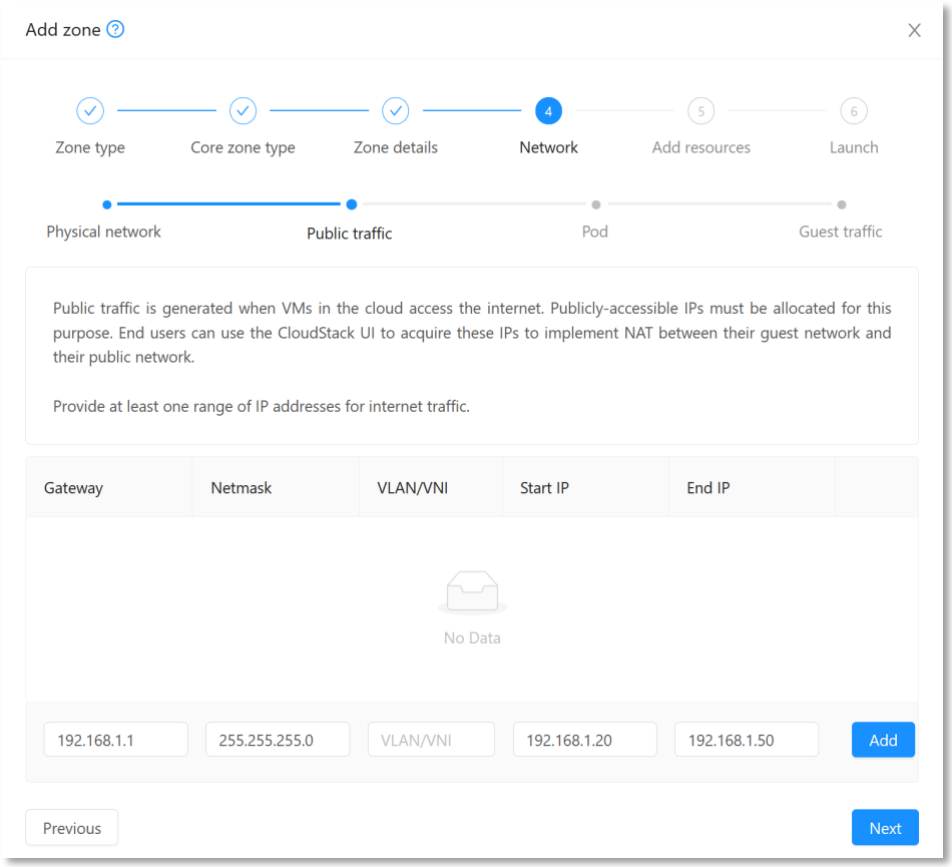
Netmask - 255.255.255.0

VLAN/VNI - leave blank

Start IP - 192.168.1.20

End IP - 192.168.1.50

Click on the “**Add**” button, click on “Next”



Pod Configuration:

Name – e.g. Pod1

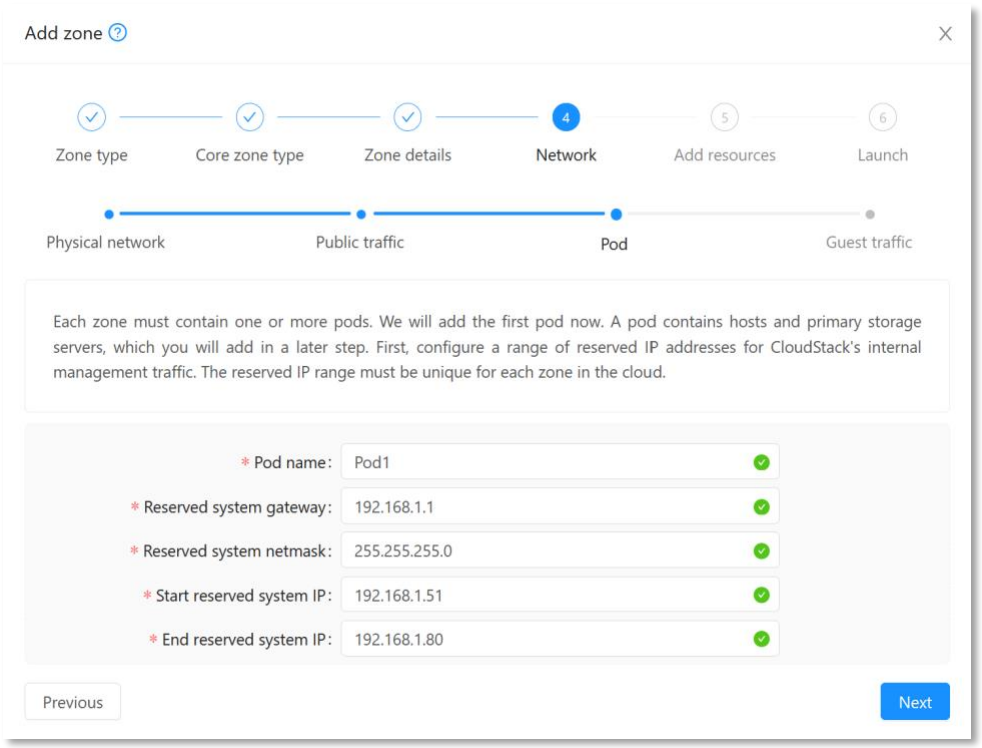
Reserved system gateway - 192.168.1.1

Reserved system netmask - 255.255.255.0

Start reserved system IP - 192.168.1.51

Start reserved system IP - 192.168.1.80

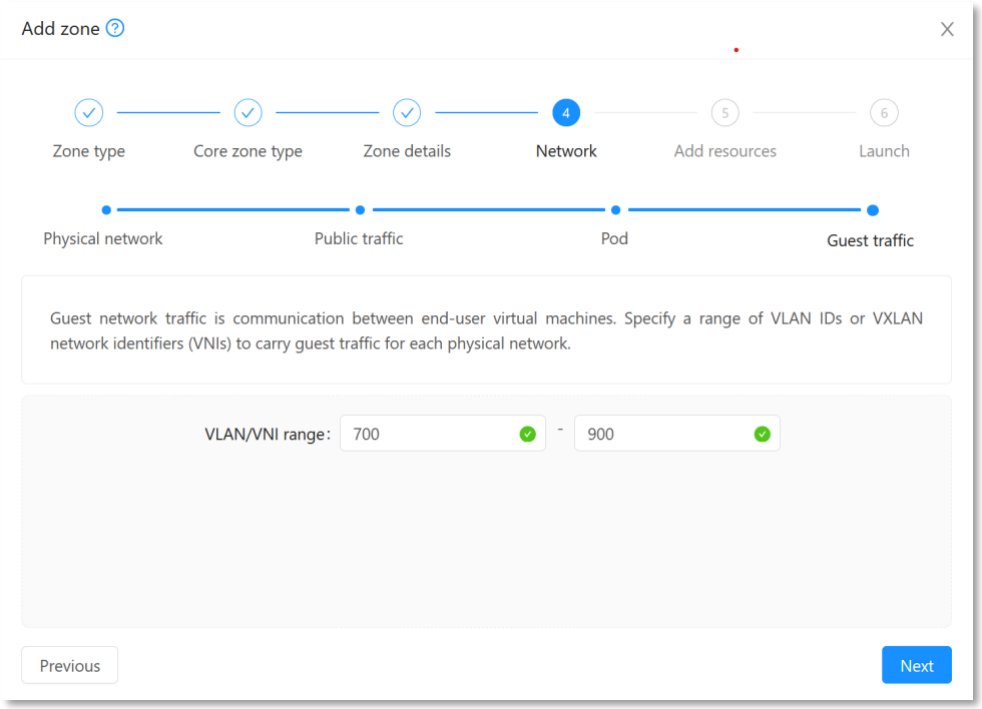
Click on “Next”



Guest traffic:

VLAN/VNI range: 700-900

Click on “Next”



Add Resources

Create a cluster with the following:

Name – e.g. Cluster1

Click on “Next”

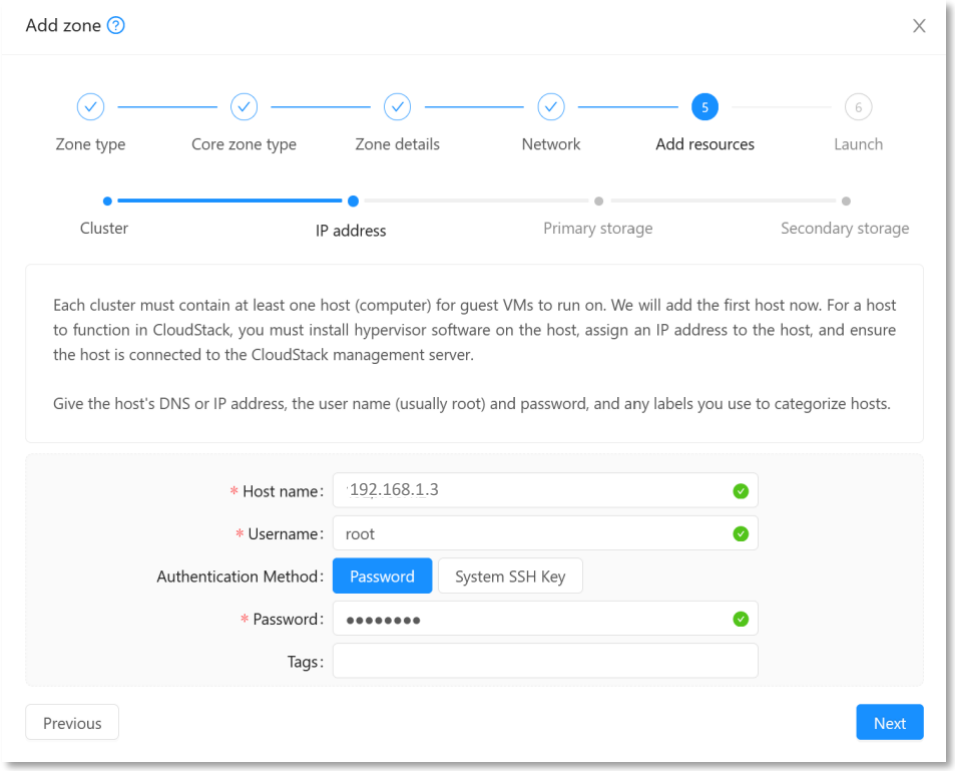
Add your default/first host:

Hostname - 192.168.1.3 (or whatever IP you have set up for this machine)

Username - root

Password - <password for root user>

Click on “Next”



Add primary storage:

Name – e.g. Primary1

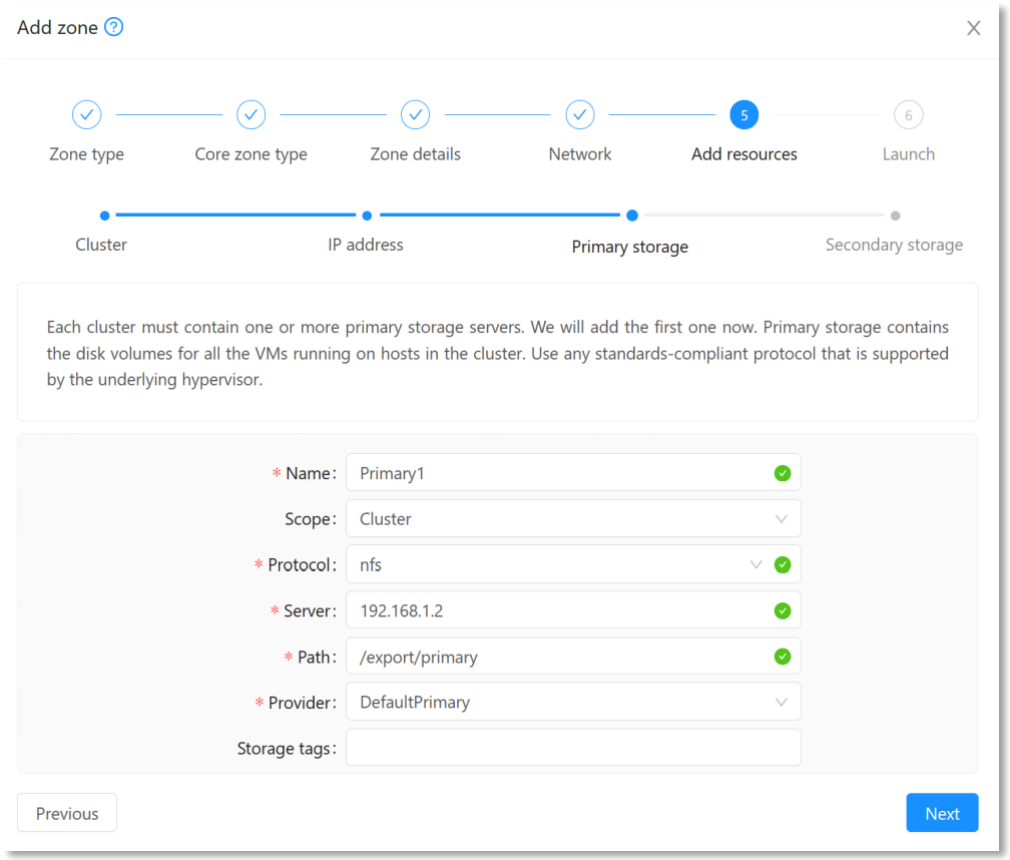
Scope - Zone

Protocol - NFS

Server - 192.168.1.2

Path - /export/primary

Click on “Next”



Add secondary storage:

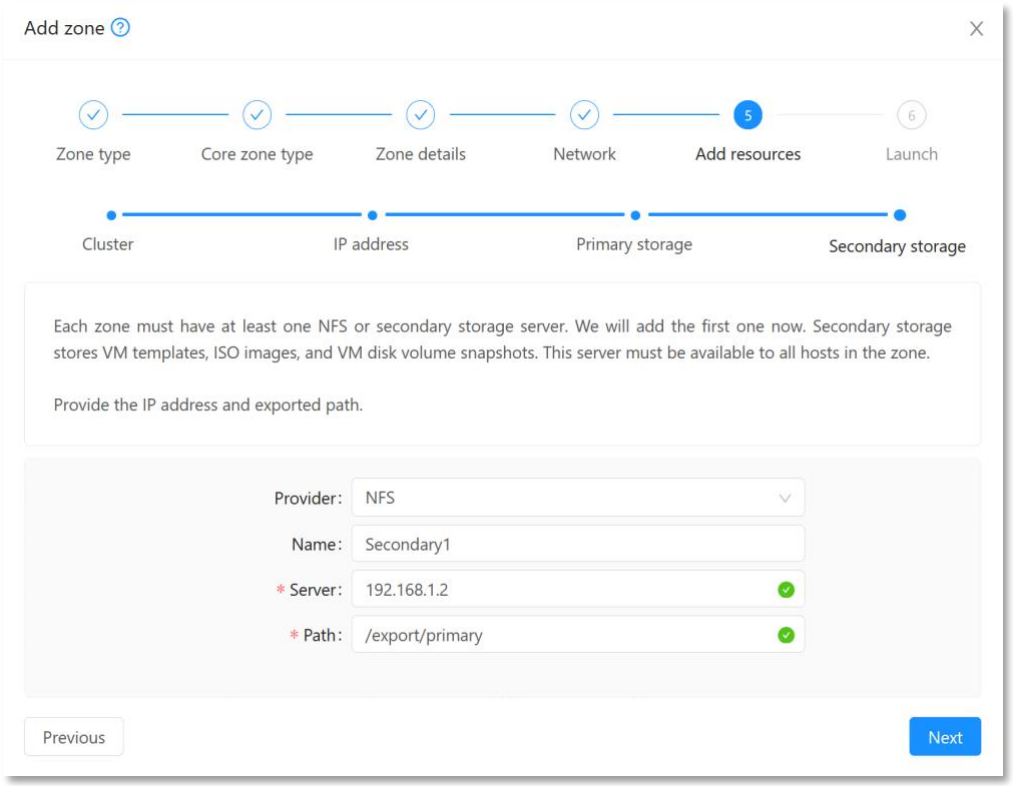
Provider - NFS

Name – e.g. Secondary1

Server - 192.168.1.2

Path - /export/secondary

Click on “Next”



Next, click “Launch zone” which will perform the following actions:

Create Zone

Create Physical networks:

- Add various traffic types to the physical network
- Update and enable the physical network
- Configure, enable and update various network providers and elements such as the virtual network element

Create Pod

Configure public traffic

Configure guest traffic (VLAN range for physical network)

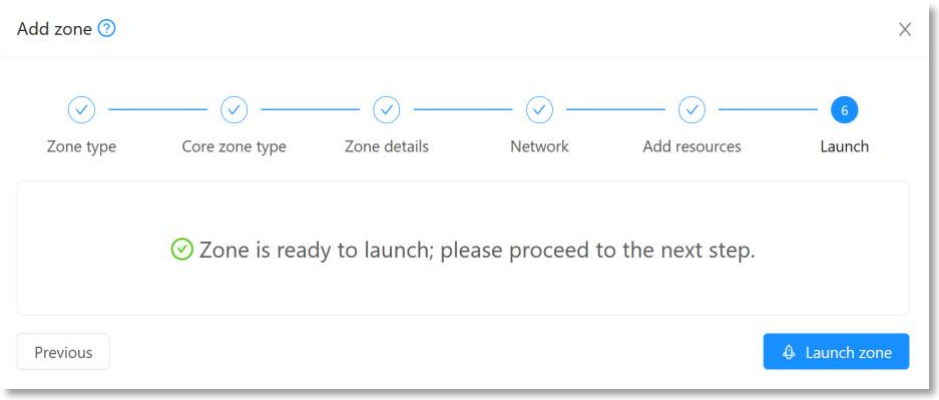
Create Cluster

Add host

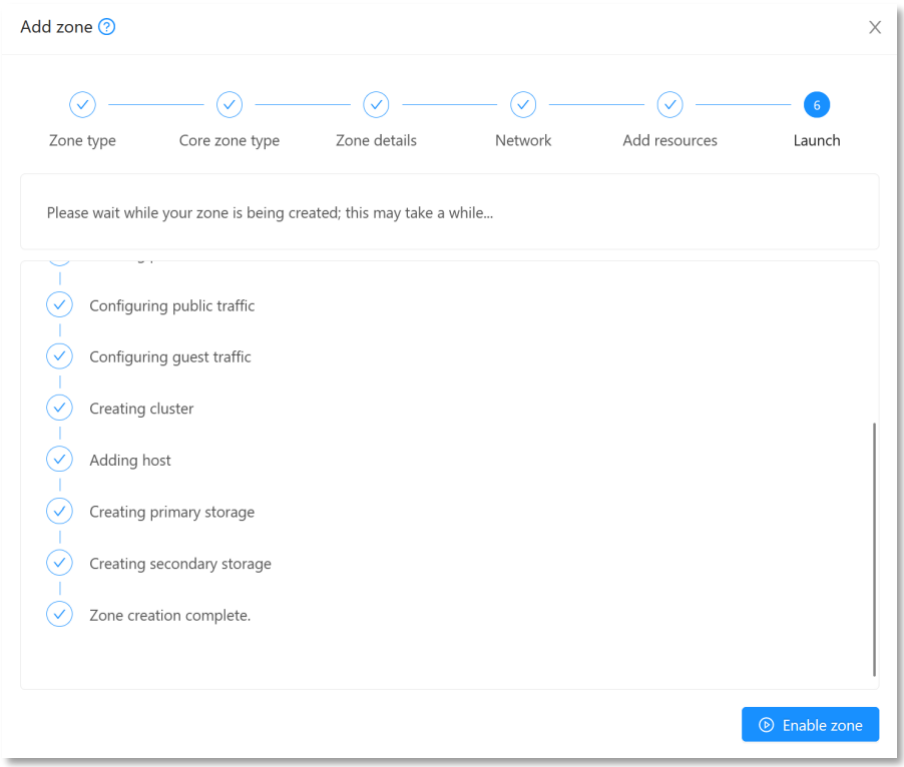
Create primary storage (also mount it on the KVM host)

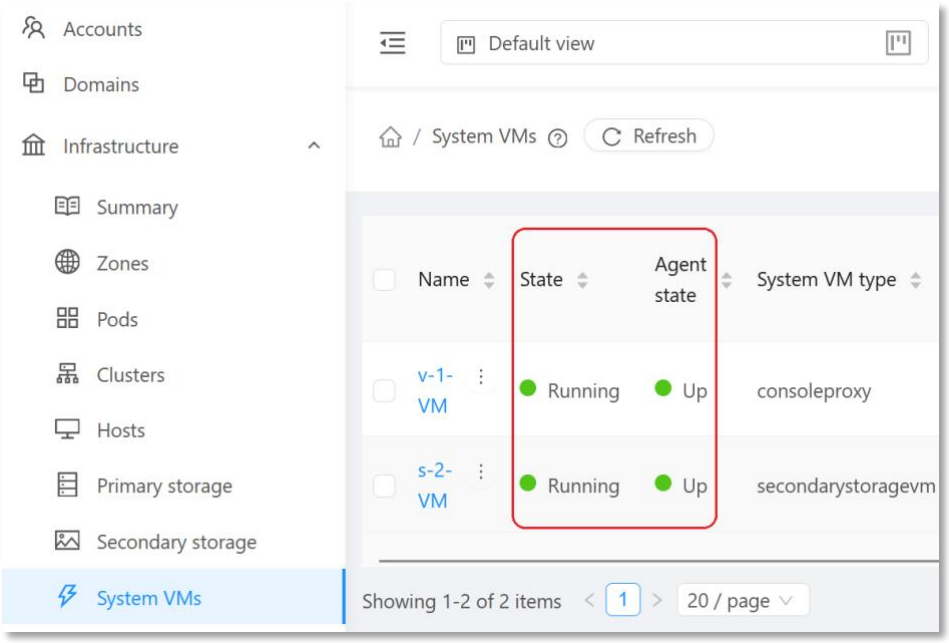
Create secondary storage

Complete zone creation



Finally, confirm and enable the zone. Wait for the system VMs to come up, and then you can proceed with your IaaS usage.





After the Secondary System Virtual Machine is up and running, you are ready to add additional hosts, download templates (and ISOs) for your IaaS platform. Just make sure those are arm64-based and you are good to go.

Conclusion

This reference architecture guide has outlined the process of deploying a CloudStack-based IaaS platform on Ampere processors, offering a scalable and efficient solution for modern cloud infrastructures. By utilising the high-performance capabilities of Ampere Altra Max processors, this setup not only optimises resource use but also significantly reduces power consumption, making it a cost-effective choice for long-term operations.

Apache CloudStack, a powerful and flexible all-in-one IaaS solution, has been proven to work seamlessly on ARM64 architecture, offering hypervisor-agnostic support and compatibility with various storage types. This adaptability makes it well-suited for on-premises, public, and hybrid cloud environments. The successful integration of CloudStack with ARM64 technology, supported by Ampere and ShapeBlue's collaboration, positions ARM64 as a strong contender in the evolving cloud landscape, providing organisations with an innovative, sustainable, and high-performing alternative to traditional x86-based systems.

Using Ampere Altra Max CPUs, you will be able to achieve a much higher CPU core density within less than or similar power envelope as of those based on x64 architecture, and lower your power consumption costs in general, while achieving the same or comparable level of performance for most workloads. Ampere provides some comparison and benchmark figures here which you can reference and decide if this is a good suit for your cloud workloads.

www.shapeblue.com

About **ShapeBlue**

ShapeBlue is the largest independent integrator of CloudStack technologies globally and are specialists in the design and implementation of IaaS cloud infrastructures for both private and public cloud implementations. We combine 100's of person-years of experience in designing and building complex network, storage and compute infrastructures with globally leading skills in Apache CloudStack.

www.amperecomputing.com

About **Ampere**

Ampere is a **modern semiconductor company** designing **the future of hyperscale cloud and edge computing** with the world's first Cloud Native Processors. Built for the **sustainable cloud with a modern 64-bit Arm server-based architecture**, Ampere works with leading cloud service providers to enable them to accelerate the delivery of all cloud computing applications. With industry-leading **cloud performance, power efficiency and scalability**, Ampere processors are tailored for the continued growth of cloud and edge computing.