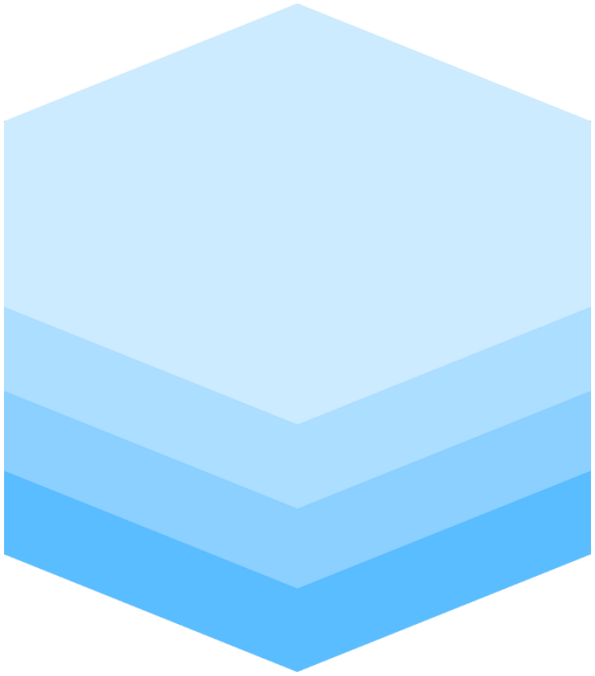
STORIDGE



ContainerIO

Quick Start Guide

ENGLISH

This manual is guides the installation and usage of ContainerIO (CIO) software. For additional information, please refer to storidge.com.

Contents

Table of Contents

Minimum System Requirements 3
Recommendations for Data Drives 3
Installing on Bare Metal and VMs 5
Download RPM and Install Script 5
Installing CIO on Centos 7 5
Installing CIO on Ubuntu 14.04 LTS 5
Installing on AWS 6
Create and Initialize Cluster 7
CIO Basics 9
Volume Creation 9
Labels 10
Profiles 11
Interacting with volumes 12
Adding nodes to cluster 12
NFS and SMB Shared Volumes 13
Using CIO with Docker 16
Docker Volume Plugin 16
Creating Data Volumes 17
CIO and Docker Swarm 20
Network Configuration 22
network.cfg file 22
Enabling jumbo frames 23

Introduction

Welcome to the Quick Start Guide for CIO (container I/O)! Storidge's CIO software was created to simplify the life of developers, DevOps and storage administrators. It provides a storage solution purpose-built for containers, making it easy to rapidly deploy applications with persistent data.

Features of CIO include:

- Rapid provisioning in seconds based on application intent
- Embedded storage orchestration making high availability simple
- Performance isolation for quality of service guarantees per volume
- Support for both stateful and stateless applications
- RESTful API and command line interface

The CIO software runs on both bare metal servers and virtual machines.

Minimum System Requirements

Bare Metal Server	Virtual Machine
2.0 GHz Duo-Core Intel x86 CPU 16GB RAM 20 GB Boot Drive and three 100 GB SSDs as data drives	2.0 GHz Duo-Core Intel x86 CPU 2GB RAM 20 GB Boot Drive and three 20 GB SSDs as data drives

Recommendations for Data Drives

During installation, CIO will automatically add drives from each node to a storage pool. A minimum of three drives per node is required to ensure data redundancy. CIO will only add raw devices to the storage pool, i.e. drives that are formatted with a file system or partitioned are not added. Delete the partitions or clear the drive metadata as needed.

Introduction, cont.

Drives with partitions can be cleared using tools such as `fdisk`. To clear drives that are mounted, first unmount and then clear the metadata area, e.g. for device sdb:

```
dd if=/dev/zero of=/dev/sdb bs=1M count=160
```

The CIO software delivers powerful but easy to use volume management across a cluster of nodes. Some of CIO's capabilities duplicate and conflict with volume management software such as LVM. Due to unpredictable results, it is highly recommended not to use two volume management software on the same drive.

System

1 GigE network
Multiple 10 GigE interfaces recommend for larger storage configurations
64-bit CentOS 7 with kernel version 3.10

CIO requires a 64-bit installation of either Centos 7 or Ubuntu 14.04. For Centos, the kernel must be 3.10, which CentOS 7 runs. For Ubuntu 14.04, the kernel used is 3.13.0. Please note the kernel version. This can be verified by using the `uname -r` command. Support for 4.x kernels will be available shortly.

As part of the installation, a request is made to optionally update your system. Staying up-to-date fixes any potential kernel-related bugs, security holes, and other issues that may have been addressed in the latest kernel packages.

TIP: You may type **`uname -r`** into console to see your kernel version. Here, the version is 3.10:

```
$ uname -r
3.10.0-327.28.3.el7.x86_64
```

Installation

Installing on Bare Metal and VMs

The installation steps are outlined below. These two steps (download and install CIO) are repeated on all nodes that will form a cluster:

1. Download the CIO RPM and install script
2. Run install script

Download RPM and Install Script

First, download the CIO package and install script on each of your nodes using your provided FTP account and `wget`. In this example, the rpm `cio-0.2.1-1720.1.x86_64.rpm` is used, but you will have a link to the latest stable release.

```
wget --user $USERNAME --password $PASSWORD \
ftp://$IP/cio-0.2.1-1720.1.x86_64.rpm

wget --user $USERNAME --password $PASSWORD ftp://$IP/install_rpm.sh
```

Installing CIO on Centos 7

Next, the install script must be executed on the RPM package. First, make the script executable:

```
chmod +x install_rpm.sh
```

Then, run the script on the downloaded RPM.

```
./install_rpm.sh cio-0.2.1-1720.1.x86_64.rpm
```

Installing CIO on Ubuntu 14.04 LTS

The Ubuntu install uses a tar package. This example uses `cio-1924-3.13.0-110-generic.amd64.tgz` but you will have a link to the latest stable release.

Download with:

```
wget --user $USERNAME --password $PASSWORD \
ftp://$IP/cio-1924-3.13.0-110-generic.amd64.tgz
```

Extract the package and install:

```
tar xvf cio-1924-3.13.0-110-generic.amd64.tgz
cd cio-1924-3.13.0-110-generic.amd64
./install
```

A containerized version of CIO is planned to simplify the installation further.

Installing on AWS

For workloads on AWS, an AMI is available with the CIO software already installed. This section provides information to consider as you review the instance launch details:

Instance Type

Amazon provides a range of general, compute, memory or storage optimized instances. Select an instance type appropriate for your workload.

Security Group

Use a security group that enables all inbound traffic from sources in the same security group.

Instance Details

Configure up to 10 instances to be launched using the CIO AMI. Some instance types enable a Placement Group option. Placement Groups provide a more consistent network performance.

Storage

Each instance must be configured with a minimum of three volumes for data redundancy. The storage used can be either ephemeral drives or EBS volumes.

Initializing Cluster

Launch the instances, then login using the key selected as part of the instance launch. Since the CIO software is already installed, proceed to the "Create and Initialize Cluster" section to prepare a cluster for use.

Create and Initialize Cluster

With the CIO software installed on all nodes, the next step is to create a cluster and then initialize the cluster for use. As part of cluster creation, CIO will automatically discover and add drive resources from each node. Note that drives that are partitioned or have a file system will not be added to the storage pool.

Start the cluster with the `cioctl create` command on the primary controller node. This will display a `cioctl join` command for adding storage nodes to the cluster, and the `cioctl init` command to complete setup of the cluster.

```
$ cioctl create

Cluster started. The current node is now the primary controller node.
To add a storage node to this cluster, run the following command:

    cioctl join 192.168.3.78 root 38513b68


After adding all storage nodes, return to this node and run following
command to initialize the cluster:

    cioctl init 38513b68
```

Single node clusters are supported. Just run the `cioctl init` command after `cioctl create`.

Multi-node clusters require a minimum of 3 nodes to ensure sufficient redundancy. In this initial release of software, a maximum of 10 nodes are supported. Copy and run the `cioctl join` command on all storage nodes to be added. Successful completion will return a message that the storage node has been added.

Initializing Cluster, cont.

```
$ cioctl join 192.168.3.78 root 38513b68

Adding this node to cluster as a storage node
```

Returning to the first node (primary node), copy and run the `cioctl init` command to start initialization of the cluster. This will take a few minutes to complete.

```
$ cioctl init 38513b68

cluster: initialization started
cluster: Copy auto-multiNode-swarm1.cfg to all nodes (NODE_NUMS:4)
cluster: Initialize target
cluster: Initialize initiator
cluster: Start node initialization
node: Clear drives
node: Load module
node: Add node backup relationship
node: Check drives
Adding disk /dev/sdd SSD to storage pool
Adding disk /dev/sdc SSD to storage pool
Adding disk /dev/sdb SSD to storage pool
Adding disk /dev/sde SSD to storage pool
.
.
.
node: Initializing metadata
cluster: Node initialization completed
cluster: Start cio daemon
cluster: Succeed: Add vd0: Type:3-copy, Size:20GB
cluster: MongoDB ready
cluster: Synchronizing VID files
cluster: Starting API
cluster: Starting plugin
```

The cluster is now ready to use.

Usage

CIO Basics

For a full list of commands, use `cio --help`. For a list of parameters available for each command, use `cio [command] --help`.

Volume Creation

Use the `cio vdadd` command to create volumes which are used by applications for persistent storage. This command takes the following format:

```
cio vdadd [flag1] [value1] [flag2] [value2] ... [flagn] [valuen]
```

Below is an example of a volume with name "foo", size 35GB, min/max IOPS of 100 and 2000, and thick provisioning:

```
$ cio vdadd --volume foo --capacity 35 --iops 100 2000 --thick
Succeed: Add vd1: Type:2-copy, Size:35GB
```

This above example creates a block device /dev/vdisk/vd1 which could be useful for applications such as databases that want to work with raw devices.

To create a volume and pass it to an application with a bind mount directory, use the `-D` or `--directory` flag:

```
$ cio vdadd --volume bar --capacity 88 -D /cio/volumes
Succeed: Add vd2: Type:2-copy, Size:88GB
```

Upon execution, CIO automatically allocates the capacity to provision the volume, format a filesystem and mount it. The filesystem is accessible through `/cio/volumes/$ID` by default, where `$ID` represents the volume number. In the above example, the `$ID` is 'vd2'.

Usage, cont.

vdadd parameters

```
-b | --bandwidth [min BW] <max BW>   set bandwidth boundaries in MB/s, cannot be used at
                                     the same time as --iops
-c | --capacity <size in GB>         vdisk size in GB
-D | --directory <directory>         set bind mount directory, defaults to /cio/volumes
-i | --iops [min IOPS] <max IOPS>    set IOPS boundaries
-l | --level <2 | 3>                 set redundancy level to 2 copy or 3 copy
-n | --node <node name>              create vdisk on named node
-N | --nodeid <node id>              create vdisk on named node
-p | --profile <profile>             use profile to add vdisk
-q | --quiet                         show vdisk ID
-T | --thick                         use thick provisioning
-v | --volume <volume name>          volume name
```

If all parameters are left blank, then a volume is created with default parameters shown below:

```
Capacity: 20GB
Provisioning: thin
Redundancy: 2-copy
Minimum IOPS: 30
Maximum IOPS: 1000000
```

Labels

CIO supports volume labels which can be useful for tagging, versioning, staging notes, licensing information, and general organization. Labels are key-value pairs. Here is an example of labels in use:

```
cio vdadd -v foo --label version=1.0 –label release=stable
```

Currently, once a label is set, it cannot be removed – only updated, using `vdmod`. Label information shows in `cio vdinfo` and `docker inspect`.

Usage, cont.

Profiles

CIO makes it really simple to manage storage through the concept of profiles. Profiles provide a way to express application intent or requirements through YAML formatted files. They can be used to consistently provision storage for applications, different classes of service, frameworks, customers, etc.

Example profiles can be viewed with `cio list`, and inspected with `cio cat $profile_name.`

Here is an example from of a profile named NGINX:

```
capacity: 25
directory: /cio/nginx
label:
 version 1.0
iops:
 min 100
 max 1000
level: 2
local: No
provision: thin
type: ssd
```

To use a profile, simply reference it using the `--profile` flag. To use the above example, we would use `cio vdadd -v foo --profile NGINX`.

To create your own, simply create a file with cio-compatible parameters in YAML format, and register it using `cio save $profile_name`. You can then confirm that your profile was saved using the `cio list` command again.

Usage, cont.

Interacting with volumes

Upon creation, a volume is instantly available for use. It can also be inspected, moved, modified or removed. Metadata about a volume is displayed with:

```
cio vdinfo –v $volume_name
```

The command `cio vdmv` moves a volume from its current node to the specified node, if the volume is not open or in use. The command format is:

```
cio vdmv -v $volume_name -n $destination_node
```

To modify the parameters of a volume, such as the capacity and the IOPS, use the command `cio vdmod`. It takes the same parameters as `vdadd`, except a target volume is mandatory.

Finally, to delete a volume, simply use `cio vdrm -v $volume_name`. You must confirm that the volume being deleted is correct. To skip the prompt, use the `-y` flag.

```
cio vdrm –y -v $volume_name
```

Adding nodes to cluster

You can scale the performance and capacity of a cluster by adding nodes. Start by running the `cioctl join-token` command on the primary (sds) node. This will display a `cioctl join` command to request addition, and the `cioctl add` command to add the new node.

```
# cioctl join-token

Issue a join request to the primary node by running the following
command on the new storage node:

    cioctl join 192.168.3.127 root 25e2b769


After the join request is received, add the new node to the cluster by
entering the following command on the primary node:

    cioctl add 25e2b769
```

Usage, cont.

NFS and SMB Shared Volumes

Volumes are normally attached to a container and the storage is not shared. There are many work flows where it is useful to have persistent storage shared by multiple containers across a cluster. CIO allows volumes to be easily created as NFS shares, using the `interface` directive in a profile. For operation within Windows environments, volumes can also be created as SMB (aka CIFS) shares.

A sample profile, named INFS is included in the rpm package. A corresponding profile, ISMB is provided for SMB shares. The following section steps through an example for an NFS share and a SMB share.

Creating a volume as a NFS share is as simple as running `cio vdadd -p INFS`. This command creates a volume and a data container that presents an NFS interface for data access. The NFS data container is allocated an IP address from a pre-defined IP address range. This ability to provision isolated volumes with separate IP address, namespace and security settings makes the NFS data container ideal for multi-tenant environments.

For example, to create a volume with an NFS interface on node swarm1:

```
$cio vdadd -p INFS -n swarm1
Succeed: Add vd1: Type:2-copy, Size:4GB
```

The example above creates a volume `vd1` with parameters specified by profile INFS, and a NFS container (instance named `nfs-vd1`) that exports volume `vd1` as a NFS share. The container can be inspected with command `docker inspect nfs-vd1`. The NFS container uses image `storidge/nfs` which exports the volume through export directory `/exports`.

To mount the NFS share, an NFS client connects to the IP address of the NFS container and mounts the export directory. For example for the instance above named `nfs-vd1`, this IP address is picked using the `--format` flag and the `docker inspect` command:

```
$docker inspect --format='{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' nfs-vd1
192.168.3.201
```

Usage, cont.

The NFS client can mount the NFS share using IP address 192.168.3.201 and export directory `/exports` on mount point `/mnt/nfs` with:

```
$mount -t nfs 192.168.3.201:/exports /mnt/nfs
```

This example assumes that the access permission for the NFS instance is set to enable access by the NFS client. The configuration and access settings for the NFS container can be set in the profile or passed through environment variable `CONFIG`.

Similarly for a SMB share, use the ISMB profile to create the volume and SMB instance:

```
$cio vdadd –p ISMB –n swarm1
Succeed: Add vd1: Type:2-copy, Size:4GB
```

And `docker inspect` to pick the IP address field. Note the naming convention for the instance has changed to `smb-vdX`:

```
$docker inspect --format='{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' smb-vd1
192.168.3.201
```

And the SMB client mounts at `/mnt/smb` with:

```
$ mount -t cifs //192.168.3.201/share /mnt/smb
```

Usage, cont.

CIO includes its own IP address management (IPAM) function. This IPAM function allocates/deallocates IP address to the NFS/SMB server containers, tracks containers as the volumes are moved to different nodes and also rebuilds the IP address map when the cluster is rebooted. The IPAM function enables the NFS/SMB server containers to keep the assigned IP address as the volume is re-scheduled to different nodes of the cluster or even after a node failover.

IP addresses managed by the IPAM function are specified in a configuration file `network.cfg`. This file is located in directory /etc/convergeio/config/. Refer to the Network Configuration section for entries in the `network.cfg` file.

TIP: To maximize data transfers for large files, you may want to enable jumbo frames in your network configuration file, e.g.:

```
$ echo MTU=9000 >> ifcfg-enp4s0
```

See Network Configuration section for additional details.

Using CIO with Docker

Docker Volume Plugin

Docker integrates with storage systems through the volume plugin API. Data volumes can be provisioned through a volume plugin and attached to an application running in a container. Since the data volume survives termination of the container, stateful data written by the application persists beyond the lifetime of the container.

The volume plugin system was introduced with Docker version 1.8.0 and continues through to version 1.12.0. As of Docker version 1.13.0, the original volume plugin is considered "legacy" as the v2 plugin architecture was introduced.

The CIO installation installs the original volume plugin (v1) for Docker version 1.12.6 and below. The new volume plugin (v2) is installed for Docker version 1.13.0 and above. The volume plugin enables requests for storage to be passed to the CIO software.

If the v2 volume plugin is not already installed, it can be downloaded and installed by using:

```
docker plugin install storidge/cio --grant-all-permissions --alias cio
```

Usage, cont.

Creating Data Volumes

In any of Docker's `--driver`, `--volume-driver`, or `--mount` parameters, the v1 or v2 volume plugin may be called as the managing software for the volume and its container. Note that the `--driver` flag is used with the `docker volume create` command. The `--volume-driver` flag is used with `docker run` command, while the `--mount flag` is used with the `docker service create` command.

Requests to create a data volume can be made both explicitly through a `docker volume create` command or implicitly via a `docker run` command.

To provision storage using the v1 volume plugin, use driver name `cio`. The v2 volume plugin is aliased to use the same `cio` driver name.

The example below works for both v1 and v2 plugin:

Create a CIO volume with name "foo" and size of 30 gigabytes:

```
docker volume create --driver cio --name foo -o capacity=30
```

Usage, cont.

The examples above specified a 30 gigabytes volume using the `--opt` or `-o` flag for volume options.

Volume options supported by the CIO plugin include:

option	description
capacity	Size in GB
directory	Host path container is bind mounted to
level	Level of data redundancy desired
type	Type of backend device
iops	Performance in min/max IOPS
bandwidth	Performance in min/max MB/s
provision	Thick or thin provisioning
profile	Template to use for volume creation

These volume options map one-to-one with the `cio vdadd` command; that is, `-o profile=NGINX` is the same as `--profile NGINX` in cio.

For example to create a volume named nginx using profile NGINX:

```
docker volume create --driver cio --name nginx -o profile=NGINX
```

To create a volume implicitly as part of `docker run` use the `--volume-driver` flag:

```
docker run --it --volume-driver cio -v foo:/tmp alpine sh
```

In this example, an Alpine image is started with a CIO volume named "foo" mounted to /tmp inside the container. If a volume named foo does not already exist, it will be automatically created with default parameters.

Usage, cont.

To list created volumes, simply run `docker volume ls`. To inspect the properties of a specific volume, use `docker volume inspect $VOLUME_NAME`. To delete them, use `docker volume rm $VOLUME_NAME.`

Usage, cont.

CIO and Docker Swarm

As of Docker 1.12. , the `--mount` flag is used to mount a data volume for a service. Docker supports two different kinds of mounts. A bind mount makes a file or directory on a host available to a container but cannot be shared with containers on other hosts. Named volumes decouples persistent data from containers and host machines.

The example below creates a named volume `mysqldb` which can be presented for access on multiple hosts across the swarm cluster as the container is rescheduled.

```
docker volume create --driver cio
--name mysqldb \
--opt size=50 \
--opt type=ssd \
--opt iops=1000,2000 \
--label version=3.3 --label release=stable
```

The named volume created as `mysqldb` can now be passed to the `docker service create` command in the `source` field, e.g.:

```
docker service create \
--mount source=mysqldb,target=/var/lib/mysql,volume-driver=cio \
--replicas 1 \
-e MYSQL_ROOT_PASSWORD=mysecret \
--name mysql \
mysql
```

Usage, cont.

Data volumes can also be created implicitly with the `docker service create` command. Unlike the `docker run` command, `docker service create` allows provisioning storage with both volume options and volume labels. This is accomplished through the `--mount` flag but with different syntax compared to `docker run` command:

```
docker service create \
--mount source=mysqldb,target=/var/lib/mysql,volume-driver=cio,volume-
opt=profile=MYSQL \
--replicas 1 \
-e MYSQL_ROOT_PASSWORD=mysecret \
--name mysql \
mysql
```

`source` is the named volume while `target` is the path within the container. This example takes advantage of the volume option field to create the named volume implicitly using profile MYSQL. The approach makes it much easier to consistently provision volumes for apps, frameworks, projects, etc. through a YAML file.

Network Configuration

network.cfg file

Docker supports the ability to create user defined networks. CIO uses this capability to create networks for shared volumes that are accessible across a cluster of nodes. The information for these user defined networks is persisted in a network configuration file `network.cfg` located in directory /etc/convergeio/config/. The network.cfg file uses a YAML format.

Each entry in file network.cfg has a network name and associated key/value pairs which specify how the network should be provisioned. When a profile containing a network name is called, CIO will check whether the network exists. If the network is not already created, CIO will check the network.cfg file and use the parameters within to create a new network.

Example of a network configuration file:

```
---
# Example network configuration file
networks:
  - name: cionet
    driver: macvlan
    iprange: 192.168.3.201-192.168.3.255
    subnet: 192.168.3.0/24
    gateway: 192.168.3.1
    port: enp4s0
```

The entry above creates a network name “cionet” using the macvlan driver. The iprange parameter specifies a range of available IP addresses on the local subnet. These IP addresses are managed by the IPAM function of CIO and are allocated to data containers on demand. The parent interface for the macvlan sub-interfaces is `enp4s0` in this example.

The network.cfg file currently supports the macvlan and overlay drivers. Data containers on a macvlan network are accessible by external NFS clients on the local network. This is useful for integrating with legacy applications or virtualized servers. The overlay driver enables NFS clients within the cluster to access the data containers.

Usage, cont.

Enabling jumbo frames

To maximize data transfers for large files, you may want to enable jumbo frames in your network configuration file. These files are located in /etc/sysconfig/network-scripts directory for CentOS. For example to enable jumbo frames for network interface enp4s0:

```
echo MTU=9000 >> /etc/sysconfig/network-scripts/ifcfg-enp4s0
```

This change should be made on both the source and destination hosts as the network drivers will always negotiate to the smallest common size. After modifying the network configuration file and rebooting, you can see a changed MTU:

```
3: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc pfifo_fast state UP
qlen 1000
    link/ether 00:25:90:65:d3:eb brd ff:ff:ff:ff:ff:ff
    inet 192.168.3.78/24 brd 192.168.3.255 scope global dynamic enp4s0
       valid_lft 84017sec preferred_lft 84017sec
```

support@storidge.com

@storidgeinc