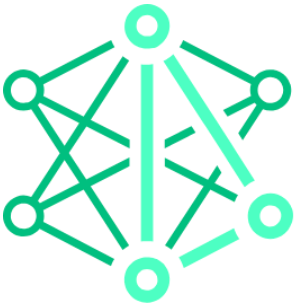

Darwin™

Darwin ™ OnPremises
Startup and Troubleshooting Guide

A SparkCognition ™ Education Document

v 1.7 - 05.2019

SparkCognition Darwin OnPremises Startup and Troubleshooting Guide

Contents

About this guide

This manual is the *Darwin™ OnPremises Startup and Troubleshooting Guide* that contains basic information to help you get started and troubleshoot Darwin issues. It is intended for data scientists, software engineers, and analysts who want to use Darwin to create and train models, monitor jobs, and perform analysis.

Documentation included with this guide:

- The *SparkCognition Darwin API User Guide*
- The *SparkCognition Darwin Python SDK Guide*

The complete SparkCognition Darwin documentation suite is available from the Darwin support portal. The documentation suite includes:

- This guide
- The *SparkCognition Darwin API User Guide*
- The *SparkCognition Darwin Python SDK Guide*
- The *Darwin Release Notes*

Darwin overview

Darwin is a SparkCognition™ tool that automates model building processes to solve specific problems. This tool enhances data scientist potential because it automates various tasks that are often manually performed. These tasks include data cleaning, latent relationship extraction, and optimal model determination. Darwin promotes rapid and accurate feature generation through both automated windowing and risk generation. Darwin quickly creates highly-accurate, dynamic models using both supervised and unsupervised learning methods.
For additional information on Darwin, contact your local SparkCognition partner for access to the white paper titled: *Darwin - A Neurogenesis Platform*.

Accessing the API

The Darwin API can normally be accessed through one of three methods:

- the Darwin Python SDK (preferred, recommended)
- the end point URL
  Note: Contact your local administrator for the Darwin API URL.
- optionally, through user-created `curl` commands

For additional information on the Darwin SDK, see the *SparkCognition Darwin Python SDK Guide*.

Starting the Darwin System

To prepare and startup the Darwin system for use, perform the following:

1. Physically install the machine
2. Connect the system to the network
3. Connect power to the system
4. Power the system on

Darwin Dependencies Tool - setup.sh

The `setup.sh` script installs all required dependencies and packages for Darwin setup:

- Installation of dependencies and packages
- Loading of environmental variables
- Extraction and loading of docker images

Perform the following to run `setup.sh`:

1. Switch to the `root` user.

   ```
   $ sudo su
   ```

2. Change directory to the `/root` directory.

   ```
   $ cd /root
   ```

3. Extract the tar file. For Non-GPU machines, enter:

   ```
   $ tar -xvzf no-gpu-darwin.tar.gz
   ```

   For GPU machines, enter:

   ```
   $ tar -xvzf gpu-darwin.tar.gz
   ```

4. Change to the newly created `deployments/darwin` directory.

   ```
   $ cd /root/deployments/darwin
   ```

5. Install Darwin dependencies by running the `setup.sh` script. For GPU machines, install Nvidia-drivers first.

   Note: Run `setup.sh` as **root** under the `/root/deployments/darwin` directory since scripts use relative paths

   ```
   $ ./bin/setup.sh
   ```

Darwin Management Tool - darwin.sh

The `darwin.sh` script provides various Darwin management functions, including:

- Installation

- Bringing containers *up* or *down*
- Generating date-limited logs for troubleshooting
- Starting and Stopping services (by name)

The `darwin.sh` script includes the following options:

- `down`
- `up`

Note: Run `darwin.sh` as **root** under the `/root/deployments/darwin` directory since scripts use relative paths

Note: Ensure the API_SERVICE_IP in `/root/deployments/darwin/resources/config/settings.cfg` is the external IP address of the server.

```
$ ./bin/darwin.sh  <option>
```

Note: For a list of Darwin containers, see the Darwin Docker Container List.

Verifying the Darwin User Interface

Make sure all containers are up and the ports are assigned for required containers (excluding `darwin_worker` containers).

Check the Swagger User Interface by navigating to: `https://<IP-ADDRESS>:8000/v1`

Check the Darwin User Interface by navigating to: `https://<IP-ADDRESS>`

Register API Key and Create Users

Register the API key for the very first time by using the Swagger UI:

1. Navigate to the `/auth/register` endpoint.
2. Copy `api_key` from the `/root/api_key` file on the Darwin server. (Do not share the api_key with anyone.)
3. Fill in `api_key`, enter the password, repeat the password and enter email.
4. Click the **Try it out!** button. This generates an access token. Copy the token. It is valid for 2 hours. If the token becomes expired, the admin can get a new token by using the `/auth/login` endpoint.

Register New Users

1. Navigate to the `/auth/register/user` endpoint.
2. Fill in the Authorization token. Prepend with `Bearer` and paste the admin's access token.

   ```
   Bearer  <admin-access-token>
   ```

3. Fill in the password, repeat the password, and enter email.
4. Click the **Try it out!** button. The user account is now created and activated.

Troubleshooting

Various troubleshooting steps are available for Darwin including password reset, manipulating containers (query, start, stop) and consulting the Darwin logs.

Docker Container Status

The following sections show basic docker container related commands that provide basic docker functionality. The commands enable you to determine if the containers are up and communicating. The commands also enable you to restart a container if directed to do so. For additional docker command information, detailed command syntax and specific examples, see the *docker docs* site.

For a list of docker containers used by Darwin, see the Darwin Docker Containers section.

Use the following basic commands to determine Docker container status:

```
$ docker info  # returns info on containers

$ docker ps -a  # shows all running containers

$ docker attach  <ContainerName>  # attach to, query
                                  # and run commands within the container
```

Note: To detach from a container and leave the container running, use the key sequence:
`CTRL-p CTRL-q`

Stop/Start/Restart Docker Containers

Use the following basic commands to `Stop/Start/Restart` Docker containers:

```
$ docker stop  <ContainerName>   #stop specific container

$ docker start  <ContainerName>   #start specific container

$ docker restart  <ContainerName>   #restart specific container
```

Darwin Logs

The troubleshooting mechanisms for Darwin include the use of Darwin logs. Consulting and analyzing the logs provides insights into system performance and behavior.

Commonly the logs are generated, bundled and sent to SparkCognition for analysis.

1. Create a compressed `tar` archive of the `logs` directory
2. Send the archive file to either SparkCognition support or your Darwin service engineer.

SparkCognition Support

SparkCognition provides support for Darwin through various resources. The following resources enable you to research issues, create a support ticket, or contact SparkCognition:

- Read the **Darwin documentation**:
  - *Darwin API User Guide*
  - *Darwin SDK Guide*
  - *Darwin FAQ*
  - *Darwin Best Practices and Tips*
  - *Darwin Release Notes*
- Create a support ticket and log issues through the **SparkCognition Support Portal**
- Send an email message to Darwin support: **darwinsupport@sparkcognition.com**
- Call Support through the support line: +1-512-400-2001

Reference

This reference section includes the following topics:

Darwin OnPremises Directory Structure

The Darwin OnPremises installation uses the following directory structure:

```
├── certs
│   ├── bundle.pem
│   ├── sparkcognition.com.key
│   ├── STAR_sparkcognition_com_bundle.crt
│   └── STAR_sparkcognition_com.crt
├── config
│   ├── containers.cfg
│   ├── docker-compose.yml
│   ├── filebeat.yml
│   ├── install.cfg
│   └── packages.cfg
├── darwin.sh
├── logs
│   ├── darwin_api.log
│   ├── graylog.log
│   ├── job_listener_1.log
│   ├── job_listener_2.log
│   ├── job_listener_3.log
│   ├── job_listener_4.log
│   └── job_listener_5.log
├── README
├── scripts
│   ├── install_tools.sh
│   ├── logging.sh
│   ├── script_tools.sh
│   ├── start_containers.sh
│   └── stop_containers.sh
└── share
```

Darwin Files

The following table shows important Darwin OnPremises files within the directory structure.

Directory	Files
--	• darwin.sh - main Darwin management tool See Darwin Management Tool • containers.zip - zipped docker containers • README
certs	Various `.crt/ and .key` files

Directory	Files
config	Contains all configuration files • containers.cfg - contains the config parameters for all services • docker-compose.yml - graylog service config • filebeat.yml - graylog config • install.cfg - specific settings, like the API key and docker log dump • packages.cfg - list of packages and bundled containers
logs	
scripts	• install_tools.sh - functions for installing Darwin • logging.sh - tools to generate and dump logs • script_tools.sh - scripting tools used in multiple places • start_containers.sh - functions to start Darwin services • stop_containers.sh - functions to stop Darwin services
share	NFS (currently not configured)

Darwin Docker Container List

Darwin requires the following Docker containers:

Component	Docker Image Name
Darwin API	`us.gcr.io/amb-dev/darwin/amb-api:darwin-1-33-0`
Job Listener	`us.gcr.io/amb-dev/amb-amb:darwin-1-7-0`
Postgres	`postgres:9.6`
mongo	`mongo:4.1`
Nginx-node-server	`us.gcr.io/amb-dev/darwin-master/nginx-node-server`
Nginx-API	`us.gcr.io/amb-dev/darwin-master/nginx-api:master`
Nginx-web	`us.gcr.io/amb-dev/darwin/nginx-web:darwin-ui-0-9-2`
Node-server	`us.gcr.io/amb-dev/darwin-master/node-server:master`

Darwin System Set Up Procedures

The basic steps required to setup the Darwin system include:

1. Update the System and Drivers

2. Install Required Packages
3. Install Optional Packages
4. Setup NFS & Firewall Rules

Optional packages for the Darwin system include:

- `htop` - a Linux process monitoring tool
- `iftop` - a Linux bandwidth monitoring tool
- `lshw` - a Linus hardware reporting tool
- `vim` - a Linux text editor

Update the System and Drivers

Perform the following to update the system and install required drivers.

Update the System

```
$ sudo yum update
```

Install wget

```
$ sudo yum install wget
```

Download and Install NVIDIA Graphics Drivers (Optional)

Note: This step is required if GPUs are present in the system.

1. Download Current Driver(s) from NVIDIA

   For example, download v396.44 of the NVIDIA driver:

   ```
   $ wget /
   http://us.download.nvidia.com/tesla/396.44/\
   nvidia-diag-driver-local-repo-rhel7-396.44-1.0-1.x86_64.rpm
   ```

2. Install NVIDIA Drivers

   ```
   $ sudo rpm -i nvidia-diag-driver-local-repo-rhel7-396.44-1.0-1.x86_64.rpm
   $ sudo yum clean all
   $ sudo yum install cuda-drivers
   ```

3. Reboot the System

   ```
   $ sudo reboot
   ```

   Note: Upon reboot, the system might require a short time to resolve the new driver(s)

NVIDIA - Additional GPU RPM

Depending on the GPU architecture (version/release), NVIDIA driver installation can fail with an error. The error commonly states that some file/package cannot be downloaded. This error can often be resolved by installing the EPEL (Extra Packages for Enterprise Linux) package.

The EPEL package provides some additional useful packages for various Enterprise Linux distributions. See the Fedora EPEL Project site for more information.

To enable the EPEL repository and use the packages, download and install the EPEL package:

1. Download the .rpm file, for example:

```
$ sudo wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

2. Install the EPEL .rpm package to enable the EPEL repository:

```
$ sudo rpm -ivh epel-release-latest-7.noarch.rpm
```

3. Verify the EPEL repository is enabled. Enter the following command:

```
$ sudo yum repolist
```

   If the repository is enabled, the output includes the EPEL repository, for example:

```
...
repo id       repo name                                          status
epel/x86_64   Extra Packages for Enterprise Linux 7 - x86_64    5,610
...
```

Install Required Packages

Install the following required packages.

Install Python3.6

1. Download the Python36u RPM

```
$ sudo yum -y install https://centos7.iuscommunity.org/ius-release.rpm
```

2. Install Python36u

```
$ sudo yum -y install python36u
```

Install pip

```
$ sudo yum install epel-release
$ sudo yum install python-pip
```

Install Docker

The Darwin implementation requires multiple Docker containers. See the Darwin Docker Containers list for more information.

1. Verify that Docker is not installed

```
$ sudo yum remove docker \
    docker-client \
    docker-client-latest \
    docker-common \
    docker-latest \
    docker-latest-logrotate \
    docker-logrotate \
    docker-selinux \
    docker-engine-selinux \
    docker-engine
```

2. Install Required Packages

```
$ sudo yum install -y yum-utils \
  device-mapper-persistent-data \
  lvm2
```

3. Setup Repository

   For example:

```
$ sudo yum-config-manager \
    --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

4. Install and Configure Docker

   a. Install Docker

```
$ sudo yum install docker-ce
```

   For additional information, see the Docker Installation Guide.

   b. Start Docker and Enable on Boot

```
$ sudo systemctl enable docker
$ sudo systemctl start docker
```

   c. Verify Docker Installation

```
$ sudo docker run hello-world
```

   Consult the output from the command. The following example response shows the installation working correctly:

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
9db2ca6ccae0: Pull complete
Digest: sha256:4b8ff392a12ed9ea17784bd3c9a8b1fa3299cac\
```

```
44aca35a85c90c5e3c7afacdc
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
```

Install Docker Compose

```
$ sudo pip install docker-compose
$ sudo yum upgrade python*
```

Install NVIDIA Docker

1. Setup Repository

   ```
   4distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
   curl -s -L https://nvidia.github.io/nvidia-docker/\
   $distribution/nvidia-docker.repo | sudo tee /etc/yum.repos.d/nvidia-docker.repo
   ```

2. Install the `nvidia-docker` Package

   ```
   $ sudo yum install nvidia-docker2
   ```

3. Reload Docker Configuration

   ```
   $ sudo pkill -SIGHUP dockerd
   ```

Install Optional Packages

The following packages are optionally recommended, depending on your requirements.

Install vim

Install the text editor `vim`

```
$ sudo yum install vim
```

Install htop

Install the Linux process monitor `htop`

```
$ sudo yum install htop
```

Install iftop

Install the Linux real time bandwidth monitor `iftop`

```
$ sudo yum install iftop
```

Install lshw

Install the Linux hardware information tool `lshw`

```
$ sudo yum install lshw
```

Setup NFS & Firewall Rules

Setup NFS

1. Install nfs-utils

   ```
   $ sudo yum -y install nfs-utils
   ```

2. Enable and Start the `nfs-server`

   ```
   $ sudo systemctl enable nfs-server.service
   $ sudo systemctl start nfs-server.service
   ```

3. Create NFS folder and Set Ownership

   ```
   $ sudo mkdir /opt/Darwin/NFS  &&  cd /opt/Darwin
   $ sudo chown nfsnobody:nfsnobody NFS
   $ sudo chmod 755 NFS
   ```

4. Add NFS Folder to Export List

   Note: In the following example, *IP* is the IP address of the client machine.

   ```
   $ sudo echo '/opt/Darwin/NFS 192.168.6.172 \
   (rw,sync,no_root_squash,no_subtree_check)' > /etc/exports
   ```

5. Export Changes

   ```
   $ sudo exportfs -a
   ```

Allow http/https Traffic Through Firewall

```
$ firewall-cmd --zone=public --permanent --add-service=https
$ firewall-cmd --zone=public --permanent --add-service=http
```

Setting Up Darwin Docker Containers

Load Docker Containers

To use the Darwin Docker containers, load them onto the machine.

Pull Docker Containers

Perform the following procedure on each Docker image to pull the containers locally.
Notes:

- The Darwin implementation requires multiple Docker containers. See the Darwin Docker Containers list for more information.
- If pulls succeed, skip forward to the Start the Containers section.

1. Pull the Docker image

```
$ docker pull  <image_name>
```

2. Save the Container

```
$ docker save --output=' <outfile> .tar'  <image_name>
```

3. Transfer the Container

   Although the following example uses `scp` to perform the file transfer, any file transfer method can be used.

```
$ scp  <outfile>  root@ <ip-address> :/opt/darwin
```

4. Load the containers

   Issue the following command on the *Docker container target machine*:

```
$ docker load --input= <outfile> .tar
```

Start the Docker Containers

Because Darwin uses multiple containers, they must be started in the following order:

1. Postgres container
2. Darwin-API container
3. Mongo container
4. nginx-API container
5. nginx-web container
6. node-server container
7. nginx-node-server container
8. Job Listener container

Start Postgres Container

```
$ docker run -d --name darwin_postgres_1 \
    --restart=unless-stopped \
    --net="darwin_default" \
    -v postgres_data:/var/lib/postgresql/data/ \
    -p 5432:5432 \
    -t ${POSTGRES_NAME}:${POSTGRES_TAG} postgres \
    -c max_connections=400 \
    -c shared_buffers=1GB \
    -c work_mem=10MB \
    -c dynamic_shared_memory_type=posix \
```

```
    -c max_files_per_process=5000 \
    -c log_destination=stderr \
    -c logging_collector=off \
    -c log_directory=pg_log \
    -c log_filename=postgresql-%Y-%m-%d_%H%M%S.log \
    -c log_file_mode=0600 \
    -c log_rotation_age=1d \
    -c log_rotation_size=10MB \
    -c log_connections=on \
    -c log_disconnections=on
```

Start Darwin-API Container

```
$ docker run -d --name darwin_api_1 \
    --shm-size 3G  \
    --restart=unless-stopped \
    --net="darwin_default" \
    --link darwin_postgres_1:postgres \
    --mount type=bind,source=${SHARED_DIR},destination="$NFS_ROOT" \
    -e "NFS_ROOT=${NFS_ROOT}" \
    -e "ROOT_API_KEY=${API_KEY}" \
    -e "GOOGLE_CLOUD_PROJECT=${GOOGLE_CLOUD_PROJECT}" \
    -e "GCLOUD_BUCKET=${GCLOUD_BUCKET}" \
    -e "GCLOUD_DATAPROC=${GCLOUD_DATAPROC}" \
    -e "GOOGLE_CLOUD_DATAPROC=${GOOGLE_CLOUD_DATAPROC}" \
    -e "LOG_LEVEL=${LOG_LEVEL}" \
    -e "SSL_PASSWORD=${SSL_PASSWORD}" \
    -e "DB_HOST=postgres" \
    -e "DB_USER=${DB_USER}" \
    -e "DB_PASS=${DB_PASS}" \
    -e "DISTRIBUTED=${DISTRIBUTED}" \
    -e "SENDGRID_API_KEY=${SENDGRID_API_KEY}" \
    -t ${DARWIN_API_NAME}:${DARWIN_API_TAG} /bin/bash -c \
     "sleep 10  && python3.6 /srv/service/amb-api/amb_api/db/run_migrations.py \
      && uwsgi -s 0.0.0.0:5000 --manage-script-name --mount /=amb_api.api:app \
     --master --chmod-socket=666 --processes 8 --threads 8"
```

Start Mongo Container

```
$ docker run -d --name darwin_mongo_1 \
    --net="darwin_default" \
    --restart=unless-stopped \
    -v mongo-data:/data/db \
    -p 27017:27017 \
    -e "MONGO_INITDB_ROOT_USERNAME=${MONGO_USER:-darwin}" \
```

```
    -e "MONGO_INITDB_ROOT_PASSWORD=${MONGO_PASSWORD:-\"\"}" \
    -t ${MONGO_NAME}:${MONGO_TAG}
```

Start nginx-API Container

```
$ docker run -d --name darwin_nginx-api_1 \
    --net="darwin_default" \
    --restart=unless-stopped \
    -v nginx-api:/srv/logs \
    --link darwin_api_1:api \
    --mount type=bind,source=/root/deployments/darwin/certs,destination=/srv/certs/ssl \
    -e "SSL_CERT_PATH=/srv/certs/ssl/cert.crt" \
    -e "SSL_KEY_PATH=/srv/certs/ssl/key.key" \
    -p 8000:443 \
    -t ${NGINX_API_NAME}:${NGINX_API_TAG}
```

Start nginx-web Container

```
$ docker run -d --name darwin_nginx-web_1 \
    --net="darwin_default" \
    --restart=unless-stopped \
    -v nginx-web:/srv/logs \
    --mount type=bind,source=/root/deployments/darwin/certs,destination=/srv/certs/ssl \
    -p 80:80 \
    -p 443:443 \
    -e "API_SERVICE_URL=https://${API_SERVICE_IP:-localhost}:8000" \
    -e "NODE_SERVER_URL=https://${API_SERVICE_IP:-localhost}:8081" \
    -e "SSL_CERT_PATH=/srv/certs/ssl/cert.crt" \
    -e "SSL_KEY_PATH=/srv/certs/ssl/key.key" \
    -t ${NGINX_WEB_NAME}:${NGINX_WEB_TAG}
```

Start node-server Container

```
$ docker run -d --name darwin_node-server_1 \
--net="darwin_default" \
--restart=unless-stopped \
--link darwin_mongo_1:mongo \
-p 8080:8080 \
-e "MONGO_URL=mongodb://${MONGO_USER:-darwin}:${MONGO_PASSWORD:-\"\"}@mongo:\
${MONGO_PORT:-27017}/${MONGO_DATABASE:-darwin}?authSource=admin" \
-t ${NODE_SERVER_NAME}:${NODE_SERVER_TAG}
```

Start nginx-node-server Container

```
$ docker run -d --name darwin_nginx-node-server_1 \
    --net="darwin_default" \
    --restart=unless-stopped \
    -p 8081:443 \
    --link darwin_node-server_1:node-server \
    -v nginx-node-server:/srv/logs \
    --mount type=bind,source=/root/deployments/darwin/certs,destination=/srv/certs/ssl \
    -e "SSL_CERT_PATH=/srv/certs/ssl/cert.crt" \
    -e "SSL_KEY_PATH=/srv/certs/ssl/key.key" \
    -t ${NGINX_NODE_SERVER_NAME}:${NGINX_NODE_SERVER_TAG}
```

Start Job Listener Container

```
$ docker run -d --name darwin_worker_$1 \
    --shm-size 3G --ulimit core=0:0 \
    --restart=unless-stopped \
    --net="darwin_default" \
    --link darwin_postgres_1:postgres \
    --mount type=bind,source="$SHARED_DIR",destination="$NFS_ROOT" \
    --entrypoint="/opt/conda/bin/listen-jobs" \
    -e "HOSTNAME=worker" \
    -e "DB_HOST=postgres" \
    -e "DB_USER=$DB_USER" \
    -e "DB_PASS=$DB_PASS" \
    -e "LOG_LEVEL=$LOG_LEVEL" \
    -e "GCLOUD_DATAPROC=$GCLOUD_DATAPROC" \
    -e "AUTOSAVE_MODELS=$AUTOSAVE_MODELS" \
    -e "SSL_PASSWORD=$SSL_PASSWORD" \
    -e "TRAIN_ASYNC=$TRAIN_ASYNC" \
    -e "USE_CUDA=$USE_CUDA" \
    -e "DISTRIBUTED=$DISTRIBUTED" \
    -e "DISTRIBUTOR_MODE=$DISTRIBUTOR_MODE" \
    -e "NFS_ROOT=$NFS_ROOT" \
    -t ${WORKER_NAME}:${WORKER_TAG}
```

Darwin Single System - Minimum Specifications

The minimum system specifications for a Darwin Single System include:

Software

Application Software	Darwin™
Operating System	Linux CentOS 7

Hardware

Processing Element	GPX XT10-22S1-6GPU

System Summary

Barebone	Darwin™
Processor	Linux CentOS 7
Memory	16 x 16GB PC4-21300 2666MHz DDR4 ECC Registered DIMM
PCI Express NVMe Drive	1.0 TB Intel® SSD DC P4500 Series U.2 PCIe 3.1 x4 NVMe Solid State Drive
Hard Drive	6 x 1.8TB SAS3 12.0Gb/s 10000RPM - 2.5” Seagate Exos 10E2400 Series (512e/4Kn)
Optical Drive	No Optical Drive Support
Co-Processors	5 x NVIDIA® Tesla™ P100 GPU Computing Accelerator - 12GB HBM2 - PCIe 3.0 x 16 Passive
Controller Card	Supermicro AOC-S3108L-H8iR-16DD SAS 3.0 12Gb/s 8-Port RAID Controller with 2G Cache
Battery Backup	CacheVault Flash Module Protection for LSI 3108 SAS Controller Mounting Kit for Flash or Battery Backup Module (PCIe Slot Kit)
I/O Modules - Networking	Supermicro SIOM 1-Gigabit Ethernet Adapter AOC-MGP-i2M (2x RJ45)

Technical Specifications (Barebone)

Memory Technology	DDR4 ECC Reg
Chipset Intel	C620
From Factor	2U
Color	Black
Memory Slots	16x 288-pin DIMM Sockets
Graphics	ASPEED AST2500 BMC
Ethernet	Intel® Dual Port Gigabit Ethernet
Power	2000W Redundant Platinum High-efficiency Power Supply
External Bays	10x 2.5” Hot -swap SATA Drive Bays
M.2	2x M.2 NVMe
Expansion Slots	6x PCI-E 3.0 x16 (double-width) slots 1x PCI-E 3.0 x16/8 Low-profle
Dimensions (WxHxD)	17.2” (437mm) x 3.5”(89mm) x 31” (787mm)

Technical Specifications (Processor)

Product Line	Xeon Scalable
Socket	LGA3647
Clock Speed	2.00 GHz
Cores/Threads	20C / 40T

Intel Virtualization Technology	Yes
Intel Hyper-Threading	Yes