1. Brief introduction

- The daemon listens on `unix:///var/run/docker.sock` but you can Bind Docker to another host/port or a Unix socket.
- The API tends to be REST. However, for some complex commands, like `attach` or `pull`, the HTTP connection is hijacked to transport `stdout`, `stdin` and `stderr`.
- A `Content-Length` header should be present in `POST` requests to endpoints that expect a body.
- To lock to a specific version of the API, you prefix the URL with the version of the API to use. For example, `/v1.18/info`. If no version is included in the URL, the maximum supported API version is used.
- If the API version specified in the URL is not supported by the daemon, a HTTP `400 Bad Request`error message is returned.

2. Errors

The Engine API uses standard HTTP status codes to indicate the success or failure of the API call. The body of the response will be JSON in the following format:

```
{
    "message": "page not found"
}
```

The status codes that are returned for each endpoint are specified in the endpoint documentation below.

3. Endpoints

3.1 Containers

LIST CONTAINERS

`GET /containers/json`

List containers

Example request:

```
GET /v1.24/containers/json?all=1&before=8dfafdbc3a40&size=1 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
[
    {
            "Id": "8dfafdbc3a40",
            "Names":["/boring_feynman"],
            "Image": "ubuntu:latest",
            "ImageID": "d74508fb6632491cea586a1fd7d748dfc5274cd6fdfedee309ecdcbc2bf5cb82",
            "Command": "echo 1",
            "Created": 1367854155,
                "State": "exited",
                "Status": "Exit 0",
                "Ports": [{"PrivatePort": 2222, "PublicPort": 3333, "Type": "tcp"}],
                "Labels": {
                        "com.example.vendor": "Acme",
                        "com.example.license": "GPL",
                        "com.example.version": "1.0"
                },
                "SizeRw": 12288,
                "SizeRootFs": 0,
                "HostConfig": {
                        "NetworkMode": "default"
                },
                "NetworkSettings": {
                        "Networks": {
                                "bridge": {
                                        "IPAMConfig": null,
                                        "Links": null,
                                        "Aliases": null,
                                        "NetworkID":
"7ea29fc1412292a2d7bba362f9253545fecdfa8ce9a6e37dd10ba8bee7129812",
                                        "EndpointID":
"2cdc4edb1ded3631c81f57966563e5c8525b81121bb3706a9a9a3ae102711f3f",
                                        "Gateway": "172.17.0.1",
                                        "IPAddress": "172.17.0.2",
                                        "IPPrefixLen": 16,
                                        "IPv6Gateway": "",
                                        "GlobalIPv6Address": "",
                                        "GlobalIPv6PrefixLen": 0,
                                        "MacAddress": "02:42:ac:11:00:02"
                                }
                        }
                },
                "Mounts": [
                        {
                                "Name": "fac362...80535",
                                "Source": "/data",
                                "Destination": "/data",
                                "Driver": "local",
                                "Mode": "ro,Z",
                                "RW": false,
                                "Propagation": ""
                        }
                ]
        },
        {
                "Id": "9cd87474be90",
                "Names":["/coolName"],
                "Image": "ubuntu:latest",
                "ImageID": "d74508fb6632491cea586a1fd7d748dfc5274cd6fdfedee309ecdcbc2bf5cb82",
                "Command": "echo 222222",
                "Created": 1367854155,
                "State": "exited",
                "Status": "Exit 0",
                "Ports": [],
                "Labels": {},
                "SizeRw": 12288,
                "SizeRootFs": 0,
                "HostConfig": {
                        "NetworkMode": "default"
                },
                "NetworkSettings": {
                        "Networks": {
                                "bridge": {
                                        "IPAMConfig": null,
                                        "Links": null,
                                        "Aliases": null,
                                        "NetworkID":
"7ea29fc1412292a2d7bba362f9253545fecdfa8ce9a6e37dd10ba8bee7129812",
                                        "EndpointID":
"88eaed7b37b38c2a3f0c4bc796494fdf51b270c2d22656412a2ca5d559a64d7a",
                                        "Gateway": "172.17.0.1",
                                        "IPAddress": "172.17.0.8",
                                        "IPPrefixLen": 16,
                                        "IPv6Gateway": "",
                                        "GlobalIPv6Address": "",
                                        "GlobalIPv6PrefixLen": 0,
                                        "MacAddress": "02:42:ac:11:00:08"
                                }
                        }
                },
                "Mounts": []
        },
        {
                "Id": "3176a2479c92",
                "Names":["/sleepy_dog"],
                "Image": "ubuntu:latest",
                "ImageID": "d74508fb6632491cea586a1fd7d748dfc5274cd6fdfedee309ecdcbc2bf5cb82",
                "Command": "echo 3333333333333333",
                "Created": 1367854154,
                "State": "exited",
                "Status": "Exit 0",
                "Ports":[],
                "Labels": {},
                "SizeRw":12288,
                "SizeRootFs":0,
                "HostConfig": {
                        "NetworkMode": "default"
                },
                "NetworkSettings": {
                        "Networks": {
                                "bridge": {
                                        "IPAMConfig": null,
                                        "Links": null,
                                        "Aliases": null,
                                        "NetworkID":
"7ea29fc1412292a2d7bba362f9253545fecdfa8ce9a6e37dd10ba8bee7129812",
                                        "EndpointID":
"8b27c041c30326d59cd6e6f510d4f8d1d570a228466f956edf7815508f78e30d",
                                        "Gateway": "172.17.0.1",
                                        "IPAddress": "172.17.0.6",
                                        "IPPrefixLen": 16,
                                        "IPv6Gateway": "",
                                        "GlobalIPv6Address": "",
                                        "GlobalIPv6PrefixLen": 0,
                                        "MacAddress": "02:42:ac:11:00:06"
                                }
                        }
                },
                "Mounts": []
        },
        {
                "Id": "4cb07b47f9fb",
                "Names":["/running_cat"],
                "Image": "ubuntu:latest",
                "ImageID": "d74508fb6632491cea586a1fd7d748dfc5274cd6fdfedee309ecdcbc2bf5cb82",
                "Command": "echo 444444444444444444444444444444444",
                "Created": 1367854152,
                "State": "exited",
                "Status": "Exit 0",
                "Ports": [],
                "Labels": {},
                "SizeRw": 12288,
                "SizeRootFs": 0,
                "HostConfig": {
                        "NetworkMode": "default"
                },
                "NetworkSettings": {
                        "Networks": {
                                "bridge": {
                                        "IPAMConfig": null,
                                        "Links": null,
                                        "Aliases": null,
                                        "NetworkID":
"7ea29fc1412292a2d7bba362f9253545fecdfa8ce9a6e37dd10ba8bee7129812",
                                        "EndpointID":
"d91c7b2f0644403d7ef3095985ea0e2370325cd2332ff3a3225c4247328e66e9",
                                        "Gateway": "172.17.0.1",
                                        "IPAddress": "172.17.0.5",
                                        "IPPrefixLen": 16,
                                        "IPv6Gateway": "",
                                        "GlobalIPv6Address": "",
                                        "GlobalIPv6PrefixLen": 0,
                                        "MacAddress": "02:42:ac:11:00:05"
                                }
                        }
                },
                "Mounts": []
        }
]
```

Query parameters:

- **all** – 1/True/true or 0/False/false, Show all containers. Only running containers are shown by default (i.e., this defaults to false)
- **limit** – Show `limit` last created containers, include non-running ones.
- **since** – Show only containers created since Id, include non-running ones.
- **before** – Show only containers created before Id, include non-running ones.
- **size** – 1/True/true or 0/False/false, Show the containers sizes
- **filters** - a JSON encoded value of the filters (a `map[string][]string`) to process on the containers list. Available filters:
- `exited=<int>`; -- containers with exit code of `<int>` ;
    -

status=(created restarting running paused exited dead)

-
- label=key or label="key=value" of a container label
-

isolation=(default process hyperv) (Windows daemon only)

-
- ancestor=(<image-name>[:<tag>], <image id> or <image@digest>)
- before=(<container id> or <container name>)
- since=(<container id> or <container name>)
- volume=(<volume name> or <mount point destination>)
- network=(<network id> or <network name>)

Status codes:

- **200** – no error
- **400** – bad parameter
- **500** – server error

CREATE A CONTAINER

POST /containers/create

Create a container

Example request:

```
POST /v1.24/containers/create HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
        "Hostname": "",
        "Domainname": "",
        "User": "",
        "AttachStdin": false,
        "AttachStdout": true,
        "AttachStderr": true,
        "Tty": false,
        "OpenStdin": false,
        "StdinOnce": false,
        "Env": [
                "FOO=bar",
                "BAZ=quux"
        ],
        "Cmd": [
                "date"
        ],
        "Entrypoint": "",
        "Image": "ubuntu",
        "Labels": {
                "com.example.vendor": "Acme",
                "com.example.license": "GPL",
                "com.example.version": "1.0"
        },
        "Volumes": {
          "/volumes/data": {}
        },
        "Healthcheck":{
           "Test": ["CMD-SHELL", "curl localhost:3000"],
           "Interval": 1000000000,
           "Timeout": 10000000000,
           "Retries": 10,
           "StartPeriod": 60000000000
        },
        "WorkingDir": "",
        "NetworkDisabled": false,
        "MacAddress": "12:34:56:78:9a:bc",
        "ExposedPorts": {
                "22/tcp": {}
        },
        "StopSignal": "SIGTERM",
        "HostConfig": {
          "Binds": ["/tmp:/tmp"],
          "Tmpfs": { "/run": "rw,noexec,nosuid,size=65536k" },
          "Links": ["redis3:redis"],
          "Memory": 0,
          "MemorySwap": 0,
          "MemoryReservation": 0,
          "KernelMemory": 0,
          "CpuPercent": 80,
          "CpuShares": 512,
          "CpuPeriod": 100000,
          "CpuQuota": 50000,
          "CpusetCpus": "0,1",
          "CpusetMems": "0,1",
          "IOMaximumBandwidth": 0,
          "IOMaximumIOps": 0,
          "BlkioWeight": 300,
          "BlkioWeightDevice": [{}],
          "BlkioDeviceReadBps": [{}],
          "BlkioDeviceReadIOps": [{}],
          "BlkioDeviceWriteBps": [{}],
          "BlkioDeviceWriteIOps": [{}],
          "MemorySwappiness": 60,
          "OomKillDisable": false,
          "OomScoreAdj": 500,
          "PidMode": "",
          "PidsLimit": -1,
          "PortBindings": { "22/tcp": [{ "HostPort": "11022" }] },
          "PublishAllPorts": false,
          "Privileged": false,
          "ReadonlyRootfs": false,
          "Dns": ["8.8.8.8"],
          "DnsOptions": [""],
          "DnsSearch": [""],
          "ExtraHosts": null,
          "VolumesFrom": ["parent", "other:ro"],
          "CapAdd": ["NET_ADMIN"],
          "CapDrop": ["MKNOD"],
          "GroupAdd": ["newgroup"],
          "RestartPolicy": { "Name": "", "MaximumRetryCount": 0 },
          "NetworkMode": "bridge",
          "Devices": [],
          "Sysctls": { "net.ipv4.ip_forward": "1" },
          "Ulimits": [{}],
          "LogConfig": { "Type": "json-file", "Config": {} },
          "SecurityOpt": [],
          "StorageOpt": {},
          "CgroupParent": "",
          "VolumeDriver": "",
          "ShmSize": 67108864
      },
      "NetworkingConfig": {
          "EndpointsConfig": {
              "isolated_nw" : {
                  "IPAMConfig": {
                      "IPv4Address":"172.20.30.33",
                      "IPv6Address":"2001:db8:abcd::3033",
                      "LinkLocalIPs":["169.254.34.68", "fe80::3468"]
                  },
                  "Links":["container_1", "container_2"],
                  "Aliases":["server_x", "server_y"]
              }
          }
```

```
        }
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
        "Id":"e90e34656806",
        "Warnings":[]
}
```

JSON parameters:

- **Hostname** - A string value containing the hostname to use for the container. This must be a valid RFC 1123 hostname.
- **Domainname** - A string value containing the domain name to use for the container.
- **User** - A string value specifying the user inside the container.
- **AttachStdin** - Boolean value, attaches to `stdin`.
- **AttachStdout** - Boolean value, attaches to `stdout`.
- **AttachStderr** - Boolean value, attaches to `stderr`.
- **Tty** - Boolean value, Attach standard streams to a `tty`, including `stdin` if it is not closed.
- **OpenStdin** - Boolean value, opens `stdin`,
- **StdinOnce** - Boolean value, close `stdin` after the 1 attached client disconnects.
- **Env** - A list of environment variables in the form of `["VAR=value", ...]`
- **Labels** - Adds a map of labels to a container. To specify a map: `{"key":"value", ... }`
- **Cmd** - Command to run specified as a string or an array of strings.
- **Entrypoint** - Set the entry point for the container as a string or an array of strings.
- **Image** - A string specifying the image name to use for the container.
- **Volumes** - An object mapping mount point paths (strings) inside the container to empty objects.
- **Healthcheck** - A test to perform to check that the container is healthy.
  - **Test** - The test to perform. Possible values are: + `{}` inherit healthcheck from image or parent image + `{"NONE"}` disable healthcheck + `{"CMD", args...}` exec arguments directly + `{"CMD-SHELL", command}` run command with system's default shell
  - **Interval** - The time to wait between checks in nanoseconds. It should be 0 or at least 1000000 (1 ms). 0 means inherit.
  - **Timeout** - The time to wait before considering the check to have hung. It should be 0 or at least 1000000 (1 ms). 0 means inherit.
  - **Retries** - The number of consecutive failures needed to consider a container as unhealthy. 0 means inherit.
  - **StartPeriod** - The time to wait for container initialization before starting health-retries countdown in nanoseconds. It should be 0 or at least 1000000 (1 ms). 0 means inherit.
- **WorkingDir** - A string specifying the working directory for commands to run in.
- **NetworkDisabled** - Boolean value, when true disables networking for the container

- **ExposedPorts** - An object mapping ports to an empty object in the form of:"ExposedPorts": { "<port>/<tcp|udp>: {}" }
- **StopSignal** - Signal to stop a container as a string or unsigned integer. SIGTERM by default.
- **HostConfig**
    - **Binds** – A list of volume bindings for this container. Each volume binding is a string in one of these forms:
        - host-src:container-dest to bind-mount a host path into the container. Both host-src, and container-dest must be an absolute path.
        - host-src:container-dest:ro to make the bind mount read-only inside the container. Both host-src, and container-dest must be an absolute path.
        - volume-name:container-dest to bind-mount a volume managed by a volume driver into the container. container-dest must be an absolute path.
        - volume-name:container-dest:ro to mount the volume read-only inside the container.container-dest must be an absolute path.
    - **Tmpfs** – A map of container directories which should be replaced by tmpfs mounts, and their corresponding mount options. A JSON object in the form { "/run": "rw,noexec,nosuid,size=65536k" }.
    - **Links** - A list of links for the container. Each link entry should be in the form of container_name:alias.
    - **Memory** - Memory limit in bytes.
    - **MemorySwap** - Total memory limit (memory + swap); set -1 to enable unlimited swap. You must use this with memory and make the swap value larger than memory.
    - **MemoryReservation** - Memory soft limit in bytes.
    - **KernelMemory** - Kernel memory limit in bytes.
    - **CpuPercent** - An integer value containing the usable percentage of the available CPUs. (Windows daemon only)
    - **CpuShares** - An integer value containing the container’s CPU Shares (ie. the relative weight vs other containers).
    - **CpuPeriod** - The length of a CPU period in microseconds.
    - **CpuQuota** - Microseconds of CPU time that the container can get in a CPU period.
    - **CpusetCpus** - String value containing the cgroups CpusetCpus to use.
    - **CpusetMems** - Memory nodes (MEMs) in which to allow execution (0-3, 0,1). Only effective on NUMA systems.
    - **IOMaximumBandwidth** - Maximum IO absolute rate in terms of IOps.
    - **IOMaximumIOps** - Maximum IO absolute rate in terms of bytes per second.
    - **BlkioWeight** - Block IO weight (relative weight) accepts a weight value between 10 and 1000.
    - **BlkioWeightDevice** - Block IO weight (relative device weight) in the form of:"BlkioWeightDevice": [{"Path": "device_path", "Weight": weight}]
    - **BlkioDeviceReadBps** - Limit read rate (bytes per second) from a device in the form of:"BlkioDeviceReadBps": [{"Path": "device_path", "Rate": rate}], for example:"BlkioDeviceReadBps": [{"Path": "/dev/sda", "Rate": "1024"}]"

- **BlkioDeviceWriteBps** - Limit write rate (bytes per second) to a device in the form of:"BlkioDeviceWriteBps": [{"Path": "device_path", "Rate": rate}], for example:"BlkioDeviceWriteBps": [{"Path": "/dev/sda", "Rate": "1024"}]"
- **BlkioDeviceReadIOps** - Limit read rate (IO per second) from a device in the form of:"BlkioDeviceReadIOps": [{"Path": "device_path", "Rate": rate}], for example:"BlkioDeviceReadIOps": [{"Path": "/dev/sda", "Rate": "1000"}]
- **BlkioDeviceWriteIOps** - Limit write rate (IO per second) to a device in the form of:"BlkioDeviceWriteIOps": [{"Path": "device_path", "Rate": rate}], for example:"BlkioDeviceWriteIOps": [{"Path": "/dev/sda", "Rate": "1000"}]
- **MemorySwappiness** - Tune a container's memory swappiness behavior. Accepts an integer between 0 and 100.
- **OomKillDisable** - Boolean value, whether to disable OOM Killer for the container or not.
- **OomScoreAdj** - An integer value containing the score given to the container in order to tune OOM killer preferences.
- **PidMode** - Set the PID (Process) Namespace mode for the container;"container:<name|id>": joins another container's PID namespace "host": use the host's PID namespace inside the container
- **PidsLimit** - Tune a container's pids limit. Set -1 for unlimited.
- **PortBindings** - A map of exposed container ports and the host port they should map to. A JSON object in the form { <port>/<protocol>: [{ "HostPort": "<port>" }] } Take note that port is specified as a string and not an integer value.
- **PublishAllPorts** - Allocates an ephemeral host port for all of a container's exposed ports. Specified as a boolean value.
- Ports are de-allocated when the container stops and allocated when the container starts. The allocated port might be changed when restarting the container.
- The port is selected from the ephemeral port range that depends on the kernel. For example, on Linux the range is defined by /proc/sys/net/ipv4/ip_local_port_range.
- **Privileged** - Gives the container full access to the host. Specified as a boolean value.
- **ReadonlyRootfs** - Mount the container's root filesystem as read only. Specified as a boolean value.
- **Dns** - A list of DNS servers for the container to use.
- **DnsOptions** - A list of DNS options
- **DnsSearch** - A list of DNS search domains
- **ExtraHosts** - A list of hostnames/IP mappings to add to the container's /etc/hosts file. Specified in the form ["hostname:IP"].
- **VolumesFrom** - A list of volumes to inherit from another container. Specified in the form <container name>[:<ro|rw>]
- **CapAdd** - A list of kernel capabilities to add to the container.
- **Capdrop** - A list of kernel capabilities to drop from the container.

   - **GroupAdd -** A list of additional groups that the container process will run as
   - **RestartPolicy** – The behavior to apply when the container exits. The value is an object with a Name property of either "always" to always restart, "unless-stopped" to restart always except when user has manually stopped the container or "on-failure" to restart only when the container exit code is non-zero. If on-failure is used, MaximumRetryCount controls the number of times to retry before giving up. The default is not to restart. (optional) An ever increasing delay (double the previous delay, starting at 100mS) is added before each restart to prevent flooding the server.
   - **UsernsMode -** Sets the usernamespace mode for the container when usernamespace remapping option is enabled. supported values are: host.
   - **NetworkMode -** Sets the networking mode for the container. Supported standard values are: bridge, host, none, and container:<name|id>. Any other value is taken as a custom network's name to which this container should connect to.
   - **Devices -** A list of devices to add to the container specified as a JSON object in the form{ "PathOnHost": "/dev/deviceName", "PathInContainer": "/dev/deviceName", "CgroupPermissions": "mrw"}
   - **Ulimits -** A list of ulimits to set in the container, specified as{ "Name": <name>, "Soft": <soft limit>, "Hard": <hard limit> }, for example:Ulimits: { "Name": "nofile", "Soft": 1024, "Hard": 2048 }
   - **Sysctls -** A list of kernel parameters (sysctls) to set in the container, specified as{ <name>: <Value> }, for example: { "net.ipv4.ip_forward": "1" }
   - **SecurityOpt**: A list of string values to customize labels for MLS systems, such as SELinux.
   - **StorageOpt**: Storage driver options per container. Options can be passed in the form{"size":"120G"}
   - **LogConfig -** Log configuration for the container, specified as a JSON object in the form{ "Type": "<driver_name>", "Config": {"key1": "val1"}}. Available types: json-file, syslog, journald, gelf, fluentd, awslogs, splunk, etwlogs, none. json-filelogging driver.
   - **CgroupParent -** Path to cgroups under which the container's cgroup is created. If the path is not absolute, the path is considered to be relative to the cgroups path of the init process. Cgroups are created if they do not already exist.
   - **VolumeDriver -** Driver that this container users to mount volumes.
   - **ShmSize -** Size of /dev/shm in bytes. The size must be greater than 0. If omitted the system uses 64MB.

Query parameters:

- **name** – Assign the specified name to the container. Must match /?[a-zA-Z0-9_-]+.

Status codes:

- **201** – no error
- **400** – bad parameter
- **404** – no such container
- **406** – impossible to attach (container not running)
- **409** – conflict
- **500** – server error

INSPECT A CONTAINER

GET /containers/(id or name)/json

Return low-level information on the container id

Example request:

```
GET /v1.24/containers/4fa6e0f0c678/json HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "AppArmorProfile": "",
        "Args": [
                "-c",
                "exit 9"
        ],
        "Config": {
                "AttachStderr": true,
                "AttachStdin": false,
                "AttachStdout": true,
                "Cmd": [
                        "/bin/sh",
                        "-c",
                        "exit 9"
                ],
                "Domainname": "",
                "Entrypoint": null,
                "Env": [
                        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
                ],
                "ExposedPorts": null,
                "Hostname": "ba033ac44011",
                "Image": "ubuntu",
                "Labels": {
                        "com.example.vendor": "Acme",
                        "com.example.license": "GPL",
                        "com.example.version": "1.0"
                },
                "MacAddress": "",
                "NetworkDisabled": false,
                "OnBuild": null,
                "OpenStdin": false,
                "StdinOnce": false,
                "Tty": false,
                "User": "",
                        "Volumes": {
                                        "/volumes/data": {}
                        },
                        "WorkingDir": "",
                        "StopSignal": "SIGTERM"
        },
        "Created": "2015-01-06T15:47:31.485331387Z",
        "Driver": "devicemapper",
        "ExecIDs": null,
        "HostConfig": {
                        "Binds": null,
                        "IOMaximumBandwidth": 0,
                        "IOMaximumIOps": 0,
                        "BlkioWeight": 0,
                        "BlkioWeightDevice": [{}],
                        "BlkioDeviceReadBps": [{}],
                        "BlkioDeviceWriteBps": [{}],
                        "BlkioDeviceReadIOps": [{}],
                        "BlkioDeviceWriteIOps": [{}],
                        "CapAdd": null,
                        "CapDrop": null,
                        "ContainerIDFile": "",
                        "CpusetCpus": "",
                        "CpusetMems": "",
                        "CpuPercent": 80,
                        "CpuShares": 0,
                        "CpuPeriod": 100000,
                        "Devices": [],
                        "Dns": null,
                        "DnsOptions": null,
                        "DnsSearch": null,
                        "ExtraHosts": null,
                        "IpcMode": "",
                        "Links": null,
                        "LxcConf": [],
                        "Memory": 0,
                        "MemorySwap": 0,
                        "MemoryReservation": 0,
                        "KernelMemory": 0,
                        "OomKillDisable": false,
                        "OomScoreAdj": 500,
                        "NetworkMode": "bridge",
                        "PidMode": "",
                        "PortBindings": {},
                        "Privileged": false,
                        "ReadonlyRootfs": false,
                        "PublishAllPorts": false,
                        "RestartPolicy": {
                                    "MaximumRetryCount": 2,
                                    "Name": "on-failure"
                        },
                        "LogConfig": {
                                    "Config": null,
                                    "Type": "json-file"
                        },
                        "SecurityOpt": null,
                        "Sysctls": {
                                  "net.ipv4.ip_forward": "1"
                        },
                        "StorageOpt": null,
                        "VolumesFrom": null,
                        "Ulimits": [{}],
                        "VolumeDriver": "",
                        "ShmSize": 67108864
            },
            "HostnamePath":
"/var/lib/docker/containers/ba033ac4401106a3b513bc9d639eee123ad78ca3616b921167cd74b20e25ed39/hostna
me",
            "HostsPath":
"/var/lib/docker/containers/ba033ac4401106a3b513bc9d639eee123ad78ca3616b921167cd74b20e25ed39/hosts",
            "LogPath":
"/var/lib/docker/containers/1eb5fabf5a03807136561b3c00adcd2992b535d624d5e18b6cdc6a6844d9767b/1eb5fa
bf5a03807136561b3c00adcd2992b535d624d5e18b6cdc6a6844d9767b-json.log",
            "Id": "ba033ac4401106a3b513bc9d639eee123ad78ca3616b921167cd74b20e25ed39",
            "Image": "04c5d3b7b0656168630d3ba35d8889bd0e9caafcaeb3004d2bfbc47e7c5d35d2",
            "MountLabel": "",
            "Name": "/boring_euclid",
            "NetworkSettings": {
                        "Bridge": "",
                        "SandboxID": "",
                        "HairpinMode": false,
                        "LinkLocalIPv6Address": "",
                        "LinkLocalIPv6PrefixLen": 0,
                        "Ports": null,
                        "SandboxKey": "",
                        "SecondaryIPAddresses": null,
                        "SecondaryIPv6Addresses": null,
                        "EndpointID": "",
                        "Gateway": "",
                        "GlobalIPv6Address": "",
                        "GlobalIPv6PrefixLen": 0,
                        "IPAddress": "",
                        "IPPrefixLen": 0,
                        "IPv6Gateway": "",
                        "MacAddress": "",
                        "Networks": {
                                "bridge": {
                                        "NetworkID":
"7ea29fc1412292a2d7bba362f9253545fecdfa8ce9a6e37dd10ba8bee7129812",
                                        "EndpointID":
"7587b82f0dada3656fda26588aee72630c6fab1536d36e394b2bfbcf898c971d",
                                        "Gateway": "172.17.0.1",
                                        "IPAddress": "172.17.0.2",
                                        "IPPrefixLen": 16,
                                        "IPv6Gateway": "",
                                        "GlobalIPv6Address": "",
                                        "GlobalIPv6PrefixLen": 0,
                                        "MacAddress": "02:42:ac:12:00:02"
                                }
                        }
                },
                "Path": "/bin/sh",
                "ProcessLabel": "",
                "ResolvConfPath":
"/var/lib/docker/containers/ba033ac4401106a3b513bc9d639eee123ad78ca3616b921167cd74b20e25ed39/resolv.
conf",
                "RestartCount": 1,
                "State": {
                        "Error": "",
                        "ExitCode": 9,
                        "FinishedAt": "2015-01-06T15:47:32.080254511Z",
                        "OOMKilled": false,
                        "Dead": false,
                        "Paused": false,
                        "Pid": 0,
                        "Restarting": false,
                        "Running": true,
                        "StartedAt": "2015-01-06T15:47:32.072697474Z",
                        "Status": "running"
```

```
            },
            "Mounts": [
                {
                    "Name": "fac362...80535",
                    "Source": "/data",
                    "Destination": "/data",
                    "Driver": "local",
                    "Mode": "ro,Z",
                    "RW": false,
                    "Propagation": ""
                }
            ]
}
```

Example request, with size information:

```
GET /v1.24/containers/4fa6e0f0c678/json?size=1 HTTP/1.1
```

Example response, with size information:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
....
"SizeRw": 0,
"SizeRootFs": 972,
....
}
```

Query parameters:

- **size** – 1/True/true or 0/False/false, return container size information. Default is false.

Status codes:

- **200** – no error
- **404** – no such container
- **500** – server error

LIST PROCESSES RUNNING INSIDE A CONTAINER

GET /containers/(id or name)/top

List processes running inside the container id. On Unix systems this is done by running the pscommand. This endpoint is not supported on Windows.

Example request:

```
GET /v1.24/containers/4fa6e0f0c678/top HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
     "Titles" : [
         "UID", "PID", "PPID", "C", "STIME", "TTY", "TIME", "CMD"
     ],
     "Processes" : [
         [
             "root", "13642", "882", "0", "17:03", "pts/0", "00:00:00", "/bin/bash"
         ],
         [
             "root", "13735", "13642", "0", "17:06", "pts/0", "00:00:00", "sleep 10"
         ]
     ]
}
```

Example request:

```
GET /v1.24/containers/4fa6e0f0c678/top?ps_args=aux HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "Titles" : [
       "USER","PID","%CPU","%MEM","VSZ","RSS","TTY","STAT","START","TIME","COMMAND"
   ]
   "Processes" : [
       [
           "root","13642","0.0","0.1","18172","3184","pts/0","Ss","17:03","0:00","/bin/bash"
       ],
       [
           "root","13895","0.0","0.0","4348","692","pts/0","S+","17:15","0:00","sleep 10"
       ]
   ],
}
```

Query parameters:

- **ps_args** – ps arguments to use (e.g., aux), defaults to -ef

Status codes:

- **200** – no error
- **404** – no such container

- **500** – server error

GET CONTAINER LOGS

GET /containers/(id or name)/logs

Get stdout and stderr logs from the container id

> **Note**: This endpoint works only for containers with the json-file or journald logging drivers.

Example request:

```
GET
/v1.24/containers/4fa6e0f0c678/logs?stderr=1&stdout=1&timestamps=1&follow=1&tail=10&since=1428990821
HTTP/1.1
```

Example response:

```
HTTP/1.1 101 UPGRADED
Content-Type: application/vnd.docker.raw-stream
Connection: Upgrade
Upgrade: tcp


{{ STREAM }}
```

Query parameters:

- **details** - 1/True/true or 0/False/false, Show extra details provided to logs. Default false.
- **follow** – 1/True/true or 0/False/false, return stream. Default false.
- **stdout** – 1/True/true or 0/False/false, show stdout log. Default false.
- **stderr** – 1/True/true or 0/False/false, show stderr log. Default false.
- **since** – UNIX timestamp (integer) to filter logs. Specifying a timestamp will only output log-entries since that timestamp. Default: 0 (unfiltered)
- **timestamps** – 1/True/true or 0/False/false, print timestamps for every log line. Default false.
- **tail** – Output specified number of lines at the end of logs: all or <number>. Default all.

Status codes:

- **101** – no error, hints proxy about hijacking
- **200** – no error, no upgrade header found
- **404** – no such container
- **500** – server error

INSPECT CHANGES ON A CONTAINER'S FILESYSTEM

GET /containers/(id or name)/changes

Inspect changes on container id's filesystem

Example request:

```
GET /v1.24/containers/4fa6e0f0c678/changes HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json

[
    {
        "Path": "/dev",
        "Kind": 0
    },
    {
        "Path": "/dev/kmsg",
        "Kind": 1
    },
    {
        "Path": "/test",
        "Kind": 1
    }
]
```

Values for `Kind`:

- `0`: Modify
- `1`: Add
- `2`: Delete

Status codes:

- **200** – no error
- **404** – no such container
- **500** – server error

EXPORT A CONTAINER

`GET /containers/(id or name)/export`

Export the contents of container `id`

Example request:

```
GET /v1.24/containers/4fa6e0f0c678/export HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/octet-stream


{{ TAR STREAM }}
```

Status codes:

- **200** – no error
- **404** – no such container
- **500** – server error

GET CONTAINER STATS BASED ON RESOURCE USAGE

GET /containers/(id or name)/stats

This endpoint returns a live stream of a container's resource usage statistics.

Example request:

```
GET /v1.24/containers/redis1/stats HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "read" : "2015-01-08T22:57:31.547920715Z",
    "pids_stats": {
        "current": 3
    },
    "networks": {
            "eth0": {
                "rx_bytes": 5338,
                "rx_dropped": 0,
                "rx_errors": 0,
                "rx_packets": 36,
                "tx_bytes": 648,
                "tx_dropped": 0,
                "tx_errors": 0,
                "tx_packets": 8
            },
            "eth5": {
                "rx_bytes": 4641,
                "rx_dropped": 0,
                "rx_errors": 0,
                "rx_packets": 26,
                "tx_bytes": 690,
                "tx_dropped": 0,
                "tx_errors": 0,
                "tx_packets": 9
            }
    },
    "memory_stats" : {
        "stats" : {
            "total_pgmajfault" : 0,
            "cache" : 0,
            "mapped_file" : 0,
            "total_inactive_file" : 0,
            "pgpgout" : 414,
            "rss" : 6537216,
            "total_mapped_file" : 0,
            "writeback" : 0,
            "unevictable" : 0,
            "pgpgin" : 477,
            "total_unevictable" : 0,
            "pgmajfault" : 0,
            "total_rss" : 6537216,
            "total_rss_huge" : 6291456,
            "total_writeback" : 0,
            "total_inactive_anon" : 0,
            "rss_huge" : 6291456,
            "hierarchical_memory_limit" : 67108864,
            "total_pgfault" : 964,
            "total_active_file" : 0,
            "active_anon" : 6537216,
            "total_active_anon" : 6537216,
            "total_pgpgout" : 414,
            "total_cache" : 0,
            "inactive_anon" : 0,
            "active_file" : 0,
            "pgfault" : 964,
            "inactive_file" : 0,
            "total_pgpgin" : 477
        },
        "max_usage" : 6651904,
        "usage" : 6537216,
        "failcnt" : 0,
        "limit" : 67108864
    },
    "blkio_stats" : {},
    "cpu_stats" : {
        "cpu_usage" : {
            "percpu_usage" : [
                8646879,
                24472255,
                36438778,
                30657443
            ],
            "usage_in_usermode" : 50000000,
            "total_usage" : 100215355,
            "usage_in_kernelmode" : 30000000
            },
            "system_cpu_usage" : 739306590000000,
            "throttling_data" : {"periods":0,"throttled_periods":0,"throttled_time":0}
        },
        "precpu_stats" : {
            "cpu_usage" : {
                "percpu_usage" : [
                    8646879,
                    24350896,
                    36438778,
                    30657443
                ],
                "usage_in_usermode" : 50000000,
                "total_usage" : 100093996,
                "usage_in_kernelmode" : 30000000
            },
            "system_cpu_usage" : 9492140000000,
            "throttling_data" : {"periods":0,"throttled_periods":0,"throttled_time":0}
        }
    }
```

The precpu_stats is the cpu statistic of previous read, which is used for calculating the cpu usage percent. It is not the exact copy of the cpu_stats field.

Query parameters:

- **stream** – 1/True/true or 0/False/false, pull stats once then disconnect. Default true.

Status codes:

- **200** – no error
- **404** – no such container
- **500** – server error

RESIZE A CONTAINER TTY

POST /containers/(id or name)/resize

Resize the TTY for container with id. The unit is number of characters. You must restart the container for the resize to take effect.

Example request:

```
POST /v1.24/containers/4fa6e0f0c678/resize?h=40&w=80 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

Query parameters:

- **h** – height of tty session
- **w** – width

Status codes:

- **200** – no error
- **404** – No such container
- **500** – Cannot resize container

START A CONTAINER

POST /containers/(id or name)/start

Start the container id

Example request:

```
POST /v1.24/containers/e90e34656806/start HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Query parameters:

- **detachKeys** – Override the key sequence for detaching a container. Format is a single character [a-Z] or ctrl-<value> where <value> is one of: a-z, @, ^, [, , or _.

Status codes:

- **204** – no error
- **304** – container already started
- **404** – no such container
- **500** – server error

STOP A CONTAINER

POST /containers/(id or name)/stop

Stop the container id

Example request:

```
POST /v1.24/containers/e90e34656806/stop?t=5 HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Query parameters:

- **t** – number of seconds to wait before killing the container

Status codes:

- **204** – no error
- **304** – container already stopped
- **404** – no such container
- **500** – server error

RESTART A CONTAINER

POST /containers/(id or name)/restart

Restart the container id

Example request:

```
POST /v1.24/containers/e90e34656806/restart?t=5 HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Query parameters:

- **t** – number of seconds to wait before killing the container

Status codes:

- **204** – no error
- **404** – no such container
- **500** – server error

KILL A CONTAINER

POST /containers/(id or name)/kill

Kill the container id

Example request:

```
POST /v1.24/containers/e90e34656806/kill HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Query parameters:

- **signal** - Signal to send to the container: integer or string like SIGINT. When not set, SIGKILL is assumed and the call waits for the container to exit.

Status codes:

- **204** – no error
- **404** – no such container
- **500** – server error

UPDATE A CONTAINER

POST /containers/(id or name)/update

Update configuration of one or more containers.

Example request:

```
POST /v1.24/containers/e90e34656806/update HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
  "BlkioWeight": 300,
  "CpuShares": 512,
  "CpuPeriod": 100000,
        "CpuQuota": 50000,
        "CpusetCpus": "0,1",
        "CpusetMems": "0",
        "Memory": 314572800,
        "MemorySwap": 514288000,
        "MemoryReservation": 209715200,
        "KernelMemory": 52428800,
        "RestartPolicy": {
           "MaximumRetryCount": 4,
           "Name": "on-failure"
        }
    }
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
       "Warnings": []
}
```

Status codes:

- **200** – no error
- **400** – bad parameter
- **404** – no such container
- **500** – server error

RENAME A CONTAINER

POST /containers/(id or name)/rename

Rename the container id to a new_name

Example request:

```
POST /v1.24/containers/e90e34656806/rename?name=new_name HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Query parameters:

- **name** – new name for the container

Status codes:

- **204** – no error
- **404** – no such container
- **409** - conflict name already assigned
- **500** – server error

PAUSE A CONTAINER

POST /containers/(id or name)/pause

Pause the container id

Example request:

```
POST /v1.24/containers/e90e34656806/pause HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Status codes:

- **204** – no error
- **404** – no such container
- **500** – server error

UNPAUSE A CONTAINER

POST /containers/(id or name)/unpause

Unpause the container id

Example request:

```
POST /v1.24/containers/e90e34656806/unpause HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Status codes:

- **204** – no error
- **404** – no such container
- **500** – server error

ATTACH TO A CONTAINER

POST /containers/(id or name)/attach

Attach to the container id

Example request:

```
POST /v1.24/containers/16253994b7c4/attach?logs=1&stream=0&stdout=1 HTTP/1.1
```

Example response:

```
HTTP/1.1 101 UPGRADED
Content-Type: application/vnd.docker.raw-stream
Connection: Upgrade
Upgrade: tcp


{{ STREAM }}
```

Query parameters:

- **detachKeys** – Override the key sequence for detaching a container. Format is a single character [a-Z] or ctrl-<value> where <value> is one of: a-z, @, ^, [, , or _.
- **logs** – 1/True/true or 0/False/false, return logs. Default false.
- **stream** – 1/True/true or 0/False/false, return stream. Default false.
- **stdin** – 1/True/true or 0/False/false, if stream=true, attach to stdin. Default false.
- **stdout** – 1/True/true or 0/False/false, if logs=true, return stdout log, if stream=true, attach to stdout. Default false.
- **stderr** – 1/True/true or 0/False/false, if logs=true, return stderr log, if stream=true, attach to stderr. Default false.

Status codes:

- **101** – no error, hints proxy about hijacking
- **200** – no error, no upgrade header found
- **400** – bad parameter
- **404** – no such container
- **409** - container is paused
- **500** – server error

Stream details:

When using the TTY setting is enabled in POST /containers/create , the stream is the raw data from the process PTY and client's stdin. When the TTY is disabled, then the stream is multiplexed to separatestdout and stderr.

The format is a **Header** and a **Payload** (frame).

HEADER

The header contains the information which the stream writes (stdout or stderr). It also contains the size of the associated frame encoded in the last four bytes (uint32).

It is encoded on the first eight bytes like this:

```
header := [8]byte{STREAM_TYPE, 0, 0, 0, SIZE1, SIZE2, SIZE3, SIZE4}
```

STREAM_TYPE can be:

- 0: stdin (is written on stdout)
- 1: stdout
- 2: stderr

SIZE1, SIZE2, SIZE3, SIZE4 are the four bytes of the uint32 size encoded as big endian.

PAYLOAD

The payload is the raw stream.

IMPLEMENTATION

The simplest way to implement the Attach protocol is the following:

```
1.   Read eight bytes.
2.   Choose `stdout` or `stderr` depending on the first byte.
3.   Extract the frame size from the last four bytes.
4.   Read the extracted size and output it on the correct output.
```

```
5.  Goto 1.
```

ATTACH TO A CONTAINER (WEBSOCKET)

GET /containers/(id or name)/attach/ws

Attach to the container id via websocket

Implements websocket protocol handshake according to RFC 6455

Example request

```
GET /v1.24/containers/e90e34656806/attach/ws?logs=0&stream=1&stdin=1&stdout=1&stderr=1 HTTP/1.1
```

Example response

```
{{ STREAM }}
```

Query parameters:

- **detachKeys** – Override the key sequence for detaching a container. Format is a single character [a-Z] or ctrl-<value> where <value> is one of: a-z, @, ^, [, , or _.
- **logs** – 1/True/true or 0/False/false, return logs. Default false.
- **stream** – 1/True/true or 0/False/false, return stream. Default false.
- **stdin** – 1/True/true or 0/False/false, if stream=true, attach to stdin. Default false.
- **stdout** – 1/True/true or 0/False/false, if logs=true, return stdout log, if stream=true, attach to stdout. Default false.
- **stderr** – 1/True/true or 0/False/false, if logs=true, return stderr log, if stream=true, attach to stderr. Default false.

Status codes:

- **200** – no error
- **400** – bad parameter
- **404** – no such container
- **500** – server error

WAIT A CONTAINER

POST /containers/(id or name)/wait

Block until container id stops, then returns the exit code

Example request:

```
POST /v1.24/containers/16253994b7c4/wait HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"StatusCode": 0}
```

Status codes:

- **200** – no error
- **404** – no such container

- **500** – server error

REMOVE A CONTAINER

DELETE /containers/(id or name)

Remove the container id from the filesystem

Example request:

```
DELETE /v1.24/containers/16253994b7c4?v=1 HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Query parameters:

- **v** – 1/True/true or 0/False/false, Remove the volumes associated to the container. Default false.
- **force** - 1/True/true or 0/False/false, Kill then remove the container. Default false.
- **link** - 1/True/true or 0/False/false, Remove the specified link associated to the container. Default false.

Status codes:

- **204** – no error
- **400** – bad parameter
- **404** – no such container
- **409** – conflict
- **500** – server error

RETRIEVING INFORMATION ABOUT FILES AND FOLDERS IN A CONTAINER

HEAD /containers/(id or name)/archive

See the description of the X-Docker-Container-Path-Stat header in the following section.

GET AN ARCHIVE OF A FILESYSTEM RESOURCE IN A CONTAINER

GET /containers/(id or name)/archive

Get a tar archive of a resource in the filesystem of container id.

Query parameters:

- **path** - resource in the container's filesystem to archive. Required.

- If not an absolute path, it is relative to the container's root directory. The resource specified by **path**must exist. To assert that the resource is expected to be a directory, **path** should end in / or /.(assuming a path separator of /). If **path** ends in /. then this indicates that only the contents of the **path** directory should be copied. A symlink is always resolved to its target.

- > **Note**: It is not possible to copy certain system files such as resources under /proc, /sys, /dev, and mounts created by the user in the container.

-

Example request:

```
GET /v1.24/containers/8cce319429b2/archive?path=/root HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/x-tar
X-Docker-Container-Path-Stat:
eyJuYW1lIjoicm9vdCIsInNpemUiOjQwOTYsIm1vZGUiOjIxNDc0ODQwOTYsIm10aW1lIjoiMjAxNC0wMi0yN1QyMDo1M
ToyM1oiLCJsaW5rVGFyZ2V0IjoiIn0=


{{ TAR STREAM }}
```

On success, a response header X-Docker-Container-Path-Stat will be set to a base64-encoded JSON object containing some filesystem header information about the archived resource. The above example value would decode to the following JSON object (whitespace added for readability):

```
{
    "name": "root",
    "size": 4096,
    "mode": 2147484096,
    "mtime": "2014-02-27T20:51:23Z",
    "linkTarget": ""}
```

A HEAD request can also be made to this endpoint if only this information is desired.

Status codes:

- **200** - success, returns archive of copied resource
- **400** - client error, bad parameter, details in JSON response body, one of:
  - must specify path parameter (**path** cannot be empty)
  - not a directory (**path** was asserted to be a directory but exists as a file)
- **404** - client error, resource not found, one of: – no such container (container id does not exist)
  - no such file or directory (**path** does not exist)
- **500** - server error

EXTRACT AN ARCHIVE OF FILES OR FOLDERS TO A DIRECTORY IN A CONTAINER

PUT /containers/(id or name)/archive

Upload a tar archive to be extracted to a path in the filesystem of container id.

Query parameters:

-

path - path to a directory in the container to extract the archive's contents into. Required.

-

If not an absolute path, it is relative to the container's root directory. The **path** resource must exist.

-
-

noOverwriteDirNonDir - If “1”, “true”, or “True” then it will be an error if unpacking the given content would cause an existing directory to be replaced with a non-directory and vice versa.

-

Example request:

```
PUT /v1.24/containers/8cce319429b2/archive?path=/vol1 HTTP/1.1
Content-Type: application/x-tar


{{ TAR STREAM }}
```

Example response:

```
HTTP/1.1 200 OK
```

Status codes:

- **200** – the content was extracted successfully
- **400** - client error, bad parameter, details in JSON response body, one of:
  - must specify path parameter (**path** cannot be empty)
  - not a directory (**path** should be a directory but exists as a file)
  - unable to overwrite existing directory with non-directory (if **noOverwriteDirNonDir**)
  - unable to overwrite existing non-directory with directory (if **noOverwriteDirNonDir**)
- **403** - client error, permission denied, the volume or container rootfs is marked as read-only.
- **404** - client error, resource not found, one of: – no such container (container id does not exist)
  - no such file or directory (**path** resource does not exist)
- **500** – server error

3.2 Images

LIST IMAGES

GET /images/json

Example request:

```
GET /v1.24/images/json?all=0 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
      "RepoTags": [
        "ubuntu:12.04",
        "ubuntu:precise",
        "ubuntu:latest"
      ],
         "Id": "8dbd9e392a964056420e5d58ca5cc376ef18e2de93b5cc90e868a1bbc8318c1c",
         "Created": 1365714795,
         "Size": 131506275,
         "VirtualSize": 131506275,
         "Labels": {}
   },
   {
         "RepoTags": [
            "ubuntu:12.10",
            "ubuntu:quantal"
         ],
         "ParentId": "27cf784147099545",
         "Id": "b750fe79269d2ec9a3c593ef05b4332b1d1a02a62b4accb2c21d589ff2f5f2dc",
         "Created": 1364102658,
         "Size": 24653,
         "VirtualSize": 180116135,
         "Labels": {
              "com.example.version": "v1"
         }
   }
]
```

Example request, with digest information:

```
GET /v1.24/images/json?digests=1 HTTP/1.1
```

Example response, with digest information:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
   {
        "Created": 1420064636,
        "Id": "4986bf8c15363d1c5d15512d5266f8777bfba4974ac56e3270e7760f6f0a8125",
        "ParentId": "ea13149945cb6b1e746bf28032f02e9b5a793523481a0a18645fc77ad53c4ea2",
        "RepoDigests": [

"localhost:5000/test/busybox@sha256:cbbf2f9a99b47fc460d422812b6a5adff7dfee951d8fa2e4a98caa0382cfbdbf"
        ],
        "RepoTags": [
           "localhost:5000/test/busybox:latest",
           "playdate:latest"
        ],
        "Size": 0,
        "VirtualSize": 2429728,
        "Labels": {}
    }
]
```

The response shows a single image Id associated with two repositories (RepoTags): localhost:5000/test/busybox: and playdate. A caller can use either of the RepoTags values localhost:5000/test/busybox:latest or playdate:latest to reference the image.

You can also use RepoDigests values to reference an image. In this response, the array has only one reference and that is to the localhost:5000/test/busybox repository; the playdate repository has no digest. You can reference this digest using the value:localhost:5000/test/busybox@sha256:cbbf2f9a99b47fc460d...

See the docker run and docker build commands for examples of digest and tag references on the command line.

Query parameters:

- **all** – 1/True/true or 0/False/false, default false
- **filters** – a JSON encoded value of the filters (a map[string][]string) to process on the images list. Available filters:
- dangling=true
- label=key or label="key=value" of an image label
- before=(<image-name>[:<tag>], <image id> or <image@digest>)
- since=(<image-name>[:<tag>], <image id> or <image@digest>)
- **filter** - only return images with the specified name

BUILD IMAGE FROM A DOCKERFILE

POST /build

Build an image from a Dockerfile

Example request:

```
POST /v1.24/build HTTP/1.1
Content-Type: application/x-tar


{{ TAR STREAM }}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"stream": "Step 1/5..."}
{"stream": "..."}
{"error": "Error...", "errorDetail": {"code": 123, "message": "Error..."}}
```

The input stream must be a tar archive compressed with one of the following algorithms: identity(no compression), gzip, bzip2, xz.

The archive must include a build instructions file, typically called Dockerfile at the archive's root.
The dockerfile parameter may be used to specify a different build instructions file. To do this, its value must be the path to the alternate build instructions file to use.
The archive may include any number of other files, which are accessible in the build context (See the ADD build command).
The Docker daemon performs a preliminary validation of the Dockerfile before starting the build, and returns an error if the syntax is incorrect. After that, each instruction is run one-by-one until the ID of the new image is output.
The build is canceled if the client drops the connection by quitting or being killed.

Query parameters:

- **dockerfile** - Path within the build context to the Dockerfile. This is ignored if remote is specified and points to an external Dockerfile.
- **t** – A name and optional tag to apply to the image in the name:tag format. If you omit the tagthe default latest value is assumed. You can provide one or more t parameters.
- **remote** – A Git repository URI or HTTP/HTTPS context URI. If the URI points to a single text file, the file's contents are placed into a file called Dockerfile and the image is built from that file. If the URI points to a tarball, the file is downloaded by the daemon and the contents therein used as the context for the build. If the URI points to a tarball and the dockerfile parameter is also specified, there must be a file with the corresponding path inside the tarball.
- **q** – Suppress verbose build output.
- **nocache** – Do not use the cache when building the image.
- **pull** - Attempt to pull the image even if an older image exists locally.
- **rm** - Remove intermediate containers after a successful build (default behavior).
- **forcerm** - Always remove intermediate containers (includes rm).
- **memory** - Set memory limit for build.
- **memswap** - Total memory (memory + swap), -1 to enable unlimited swap.
- **cpushares** - CPU shares (relative weight).
- **cpusetcpus** - CPUs in which to allow execution (e.g., 0-3, 0,1).
- **cpuperiod** - The length of a CPU period in microseconds.
- **cpuquota** - Microseconds of CPU time that the container can get in a CPU period.
- **buildargs** – JSON map of string pairs for build-time variables. Users pass these values at build-time. Docker uses the buildargs as the environment context for command(s) run via the Dockerfile's RUN instruction or for variable expansion in other Dockerfile instructions. This is not meant for passing secret values. Read more about the buildargs instruction
- **shmsize** - Size of /dev/shm in bytes. The size must be greater than 0. If omitted the system uses 64MB.
- **labels** – JSON map of string pairs for labels to set on the image.

Request Headers:

- **Content-type** – Set to "application/x-tar".
-

  X-Registry-Config – A base64-url-safe-encoded Registry Auth Config JSON object with the following structure:

-

```
{
    "docker.example.com": {
        "username": "janedoe",
        "password": "hunter2"
    },
    "https://index.docker.io/v1/": {
        "username": "mobydock",
        "password": "conta1n3rize14"
    }
}
```

-

This object maps the hostname of a registry to an object containing the “username” and “password” for that registry. Multiple registries may be specified as the build may be based on an image requiring authentication to pull from any arbitrary registry. Only the registry domain name (and port if not the default “443”) are required. However (for legacy reasons) the “official” Docker, Inc. hosted registry must be specified with both a “https://” prefix and a “/v1/” suffix even though Docker will prefer to use the v2 registry API.

-

Status codes:

- **200** – no error
- **500** – server error

CREATE AN IMAGE

POST /images/create

Create an image either by pulling it from the registry or by importing it

Example request:

```
POST /v1.24/images/create?fromImage=busybox&tag=latest HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"status": "Pulling..."}
{"status": "Pulling", "progress": "1 B/ 100 B", "progressDetail": {"current": 1, "total": 100}}
{"error": "Invalid..."}
...
```

When using this endpoint to pull an image from the registry, the X-Registry-Auth header can be used to include a base64-encoded AuthConfig object.

Query parameters:

- **fromImage** – Name of the image to pull. The name may include a tag or digest. This parameter may only be used when pulling an image. The pull is cancelled if the HTTP connection is closed.
- **fromSrc** – Source to import. The value may be a URL from which the image can be retrieved or -to read the image from the request body. This parameter may only be used when importing an image.
- **repo** – Repository name given to an image when it is imported. The repo may include a tag. This parameter may only be used when importing an image.
- **tag** – Tag or digest. If empty when pulling an image, this causes all tags for the given image to be pulled.

Request Headers:

- **X-Registry-Auth** – base64-encoded AuthConfig object, containing either login information, or a token
    - Credential based login:

```
{
        "username": "jdoe",
        "password": "secret",
        "email": "jdoe@acme.com"
}
```

    - Token based login:

```
{
        "identitytoken": "9cbaf023786cd7..."
}
```

Status codes:

- **200** – no error
- **404** - repository does not exist or no read access
- **500** – server error

INSPECT AN IMAGE

GET /images/(name)/json

Return low-level information on the image name

Example request:

```
GET /v1.24/images/example/json HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
    "Id" : "sha256:85f05633ddc1c50679be2b16a0479ab6f7637f8884e0cfe0f4d20e1ebb3d6e7c",
    "Container" : "cb91e48a60d01f1e27028b4fc6819f4f290b3cf12496c8176ec714d0d390984a",
    "Comment" : "",
    "Os" : "linux",
    "Architecture" : "amd64",
    "Parent" : "sha256:91e54dfb11794fad694460162bf0cb0a4fa710cfa3f60979c177d920813e267c",
    "ContainerConfig" : {
        "Tty" : false,
        "Hostname" : "e611e15f9c9d",
        "Volumes" : null,
        "Domainname" : "",
        "AttachStdout" : false,
        "PublishService" : "",
        "AttachStdin" : false,
        "OpenStdin" : false,
        "StdinOnce" : false,
        "NetworkDisabled" : false,
        "OnBuild" : [],
        "Image" : "91e54dfb11794fad694460162bf0cb0a4fa710cfa3f60979c177d920813e267c",
        "User" : "",
        "WorkingDir" : "",
        "Entrypoint" : null,
        "MacAddress" : "",
        "AttachStderr" : false,
        "Labels" : {
            "com.example.license" : "GPL",
            "com.example.version" : "1.0",
            "com.example.vendor" : "Acme"
        },
        "Env" : [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
        ],
        "ExposedPorts" : null,
        "Cmd" : [
            "/bin/sh",
            "-c",
            "#(nop) LABEL com.example.vendor=Acme com.example.license=GPL com.example.version=1.0"
        ]
    },
    "DockerVersion" : "1.9.0-dev",
    "VirtualSize" : 188359297,
    "Size" : 0,
    "Author" : "",
    "Created" : "2015-09-10T08:30:53.26995814Z",
    "GraphDriver" : {
        "Name" : "aufs",
        "Data" : null
    },
    "RepoDigests" : [

"localhost:5000/test/busybox/example@sha256:cbbf2f9a99b47fc460d422812b6a5adff7dfee951d8fa2e4a98caa0382
cfbdbf"
    ],
    "RepoTags" : [
        "example:1.0",
        "example:latest",
        "example:stable"
    ],
    "Config" : {
        "Image" : "91e54dfb11794fad694460162bf0cb0a4fa710cfa3f60979c177d920813e267c",
        "NetworkDisabled" : false,
        "OnBuild" : [],
        "StdinOnce" : false,
        "PublishService" : "",
        "AttachStdin" : false,
        "OpenStdin" : false,
        "Domainname" : "",
        "AttachStdout" : false,
        "Tty" : false,
        "Hostname" : "e611e15f9c9d",
        "Volumes" : null,
        "Cmd" : [
            "/bin/bash"
        ],
        "ExposedPorts" : null,
        "Env" : [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
        ],
        "Labels" : {
            "com.example.vendor" : "Acme",
            "com.example.version" : "1.0",
            "com.example.license" : "GPL"
        },
        "Entrypoint" : null,
            "MacAddress" : "",
            "AttachStderr" : false,
            "WorkingDir" : "",
            "User" : ""
        },
        "RootFS": {
            "Type": "layers",
            "Layers": [
                "sha256:1834950e52ce4d5a88a1bbd131c537f4d0e56d10ff0dd69e66be3b7dfa9df7e6",
                "sha256:5f70bf18a086007016e948b04aed3b82103a36bea41755b6cddfaf10ace3c6ef"
            ]
        }
}
```

Status codes:

- **200** – no error
- **404** – no such image
- **500** – server error

GET THE HISTORY OF AN IMAGE

GET /images/(name)/history

Return the history of the image name

Example request:

```
GET /v1.24/images/ubuntu/history HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "Id": "3db9c44f45209632d6050b35958829c3a2aa256d81b9a7be45b362ff85c54710",
        "Created": 1398108230,
        "CreatedBy": "/bin/sh -c #(nop) ADD
file:eb15dbd63394e063b805a3c32ca7bf0266ef64676d5a6fab4801f2e81e2a5148 in /",
        "Tags": [
            "ubuntu:lucid",
            "ubuntu:10.04"
        ],
        "Size": 182964289,
        "Comment": ""
    },
    {
            "Id": "6cfa4d1f33fb861d4d114f43b25abd0ac737509268065cdfd69d544a59c85ab8",
            "Created": 1398108222,
            "CreatedBy": "/bin/sh -c #(nop) MAINTAINER Tianon Gravi <admwiggin@gmail.com> -
mkimage-debootstrap.sh -i iproute,iputils-ping,ubuntu-minimal -t lucid.tar.xz lucid
http://archive.ubuntu.com/ubuntu/",
            "Tags": null,
            "Size": 0,
            "Comment": ""
      },
      {
            "Id": "511136ea3c5a64f264b78b5433614aec563103b4d4702f3ba7d4d2698e22c158",
            "Created": 1371157430,
            "CreatedBy": "",
            "Tags": [
                  "scratch12:latest",
                  "scratch:latest"
            ],
            "Size": 0,
            "Comment": "Imported from -"
      }
]
```

Status codes:

- **200** – no error
- **404** – no such image
- **500** – server error

PUSH AN IMAGE ON THE REGISTRY

POST /images/(name)/push

Push the image name on the registry

Example request:

```
POST /v1.24/images/test/push HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"status": "Pushing..."}
{"status": "Pushing", "progress": "1/? (n/a)", "progressDetail": {"current": 1}}}
{"error": "Invalid..."}
...
```

If you wish to push an image on to a private registry, that image must already have a tag into a repository which references that registry hostname and port. This repository name should then be used in the URL. This duplicates the command line's flow.

The push is cancelled if the HTTP connection is closed.

Example request:

```
POST /v1.24/images/registry.acme.com:5000/test/push HTTP/1.1
```

Query parameters:

- **tag** – The tag to associate with the image on the registry. This is optional.

Request Headers:

- **X-Registry-Auth** – base64-encoded AuthConfig object, containing either login information, or a token
    - Credential based login:
    -
    ```
    {
            "username": "jdoe",
            "password": "secret",
            "email": "jdoe@acme.com",
    }
    ```
    -
    - Identity token based login:
    -
    ```
    {
            "identitytoken": "9cbaf023786cd7..."
    }
    ```
    -

Status codes:

- **200** – no error
- **404** – no such image
- **500** – server error

TAG AN IMAGE INTO A REPOSITORY

POST /images/(name)/tag

Tag the image name into a repository

Example request:

```
POST /v1.24/images/test/tag?repo=myrepo&tag=v42 HTTP/1.1
```

Example response:

```
HTTP/1.1 201 Created
```

Query parameters:

- **repo** – The repository to tag in
- **tag** - The new tag name

Status codes:

- **201** – no error
- **400** – bad parameter
- **404** – no such image
- **409** – conflict
- **500** – server error

REMOVE AN IMAGE

DELETE /images/(name)

Remove the image name from the filesystem

Example request:

```
DELETE /v1.24/images/test HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-type: application/json

[
 {"Untagged": "3e2f21a89f"},
 {"Deleted": "3e2f21a89f"},
 {"Deleted": "53b4f83ac9"}
]
```

Query parameters:

- **force** – 1/True/true or 0/False/false, default false
- **noprune** – 1/True/true or 0/False/false, default false

Status codes:

- **200** – no error
- **404** – no such image
- **409** – conflict
- **500** – server error

SEARCH IMAGES

GET /images/search

Search for an image on Docker Hub.

> **Note**: The response keys have changed from API v1.6 to reflect the JSON sent by the registry server to the docker daemon's request.

Example request:

```
GET /v1.24/images/search?term=sshd HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
        {
            "description": "",
            "is_official": false,
            "is_automated": false,
            "name": "wma55/u1210sshd",
            "star_count": 0
        },
        {
            "description": "",
            "is_official": false,
            "is_automated": false,
            "name": "jdswinbank/sshd",
            "star_count": 0
        },
        {
            "description": "",
            "is_official": false,
            "is_automated": false,
            "name": "vgauthier/sshd",
            "star_count": 0
        }
...
]
```

Query parameters:

- **term** – term to search
- **limit** – maximum returned search results
- **filters** – a JSON encoded value of the filters (a map[string][]string) to process on the images list. Available filters:
- stars=<number>
- is-automated=(true|false)
- is-official=(true|false)

Status codes:

- **200** – no error
- **500** – server error

3.3 Misc

CHECK AUTH CONFIGURATION

POST /auth

Validate credentials for a registry and get identity token, if available, for accessing the registry without password.

Example request:

```
POST /v1.24/auth HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
        "username": "hannibal",
        "password": "xxxx",
        "serveraddress": "https://index.docker.io/v1/"
}
```

Example response:

```
HTTP/1.1 200 OK

{
        "Status": "Login Succeeded",
        "IdentityToken": "9cbaf023786cd7..."
}
```

Status codes:

- **200** – no error
- **204** – no error
- **500** – server error

DISPLAY SYSTEM-WIDE INFORMATION

GET /info

Display system-wide information

Example request:

```
GET /v1.24/info HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
      "Architecture": "x86_64",
      "ClusterStore": "etcd://localhost:2379",
      "CgroupDriver": "cgroupfs",
"Containers": 11,
"ContainersRunning": 7,
"ContainersStopped": 3,
"ContainersPaused": 1,
"CpuCfsPeriod": true,
"CpuCfsQuota": true,
"Debug": false,
"DockerRootDir": "/var/lib/docker",
"Driver": "btrfs",
"DriverStatus": [[""]],
"ExperimentalBuild": false,
"HttpProxy": "http://test:test@localhost:8080",
"HttpsProxy": "https://test:test@localhost:8080",
"ID": "7TRN:IPZB:QYBB:VPBQ:UMPP:KARE:6ZNR:XE6T:7EWV:PKF4:ZOJD:TPYS",
"IPv4Forwarding": true,
"Images": 16,
"IndexServerAddress": "https://index.docker.io/v1/",
"InitPath": "/usr/bin/docker",
"InitSha1": "",
"KernelMemory": true,
"KernelVersion": "3.12.0-1-amd64",
"Labels": [
    "storage=ssd"
],
"MemTotal": 2099236864,
"MemoryLimit": true,
"NCPU": 1,
"NEventsListener": 0,
"NFd": 11,
"NGoroutines": 21,
"Name": "prod-server-42",
"NoProxy": "9.81.1.160",
"OomKillDisable": true,
"OSType": "linux",
"OperatingSystem": "Boot2Docker",
"Plugins": {
    "Volume": [
        "local"
    ],
    "Network": [
        "null",
        "host",
        "bridge"
```

```
            ]
        },
        "RegistryConfig": {
            "IndexConfigs": {
                "docker.io": {
                    "Mirrors": null,
                    "Name": "docker.io",
                    "Official": true,
                    "Secure": true
                }
            },
            "InsecureRegistryCIDRs": [
                "127.0.0.0/8"
            ]
        },
        "SecurityOptions": [
            "apparmor",
            "seccomp",
            "selinux"
        ],
        "ServerVersion": "1.9.0",
        "SwapLimit": false,
        "SystemStatus": [["State", "Healthy"]],
        "SystemTime": "2015-03-10T11:11:23.730591467-07:00"
}
```

Status codes:

- **200** – no error
- **500** – server error

SHOW THE DOCKER VERSION INFORMATION

GET /version

Show the docker version information

Example request:

```
GET /v1.24/version HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "Version": "1.12.0",
        "Os": "linux",
        "KernelVersion": "3.19.0-23-generic",
        "GoVersion": "go1.6.3",
        "GitCommit": "deadbee",
        "Arch": "amd64",
        "ApiVersion": "1.24",
        "BuildTime": "2016-06-14T07:09:13.444803460+00:00",
        "Experimental": true
}
```

Status codes:

- **200 –** no error
- **500 –** server error

PING THE DOCKER SERVER

GET /_ping

Ping the docker server

Example request:

```
GET /v1.24/_ping HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: text/plain

OK
```

Status codes:

- **200** - no error
- **500** - server error

CREATE A NEW IMAGE FROM A CONTAINER’S CHANGES

POST /commit

Create a new image from a container’s changes

Example request:

```
POST /v1.24/commit?container=44c004db4b17&comment=message&repo=myrepo HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
        "Hostname": "",
        "Domainname": "",
        "User": "",
        "AttachStdin": false,
        "AttachStdout": true,
        "AttachStderr": true,
        "Tty": false,
        "OpenStdin": false,
        "StdinOnce": false,
        "Env": null,
        "Cmd": [
                "date"
        ],
        "Mounts": [
          {
            "Source": "/data",
            "Destination": "/data",
            "Mode": "ro,Z",
            "RW": false
          }
        ],
        "Labels": {
                "key1": "value1",
                "key2": "value2"
         },
        "WorkingDir": "",
        "NetworkDisabled": false,
        "ExposedPorts": {
                "22/tcp": {}
        }
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{"Id": "596069db4bf5"}
```

JSON parameters:

- **config -** the container's configuration

Query parameters:

- **container** – source container
- **repo** – repository
- **tag** – tag
- **comment** – commit message
- **author** – author (e.g., "John Hannibal Smith <hannibal@a-team.com>")
- **pause** – 1/True/true or 0/False/false, whether to pause the container before committing
- **changes** – Dockerfile instructions to apply while committing

Status codes:

- **201** – no error
- **404** – no such container
- **500** – server error

MONITOR DOCKER'S EVENTS

GET /events

Get container events from docker, in real time via streaming.

Docker containers report the following events:

```
attach, commit, copy, create, destroy, detach, die, exec_create, exec_detach, exec_start, export, health_status, kill, oom,
pause, rename, resize, restart, start, stop, top, unpause, update
```

Docker images report the following events:

```
delete, import, load, pull, push, save, tag, untag
```

Docker volumes report the following events:

```
create, mount, unmount, destroy
```

Docker networks report the following events:

```
create, connect, disconnect, destroy
```

Docker daemon report the following event:

```
reload
```

Example request:

```
GET /v1.24/events?since=1374067924
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Server: Docker/1.12.0 (linux)
Date: Fri, 29 Apr 2016 15:18:06 GMT
Transfer-Encoding: chunked

{
  "status": "pull",
  "id": "alpine:latest",
  "Type": "image",
  "Action": "pull",
  "Actor": {
    "ID": "alpine:latest",
    "Attributes": {
      "name": "alpine"
    }
    },
    "time": 1461943101,
    "timeNano": 1461943101301854122
}
{
    "status": "create",
    "id": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
    "from": "alpine",
    "Type": "container",
    "Action": "create",
    "Actor": {
        "ID": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
        "Attributes": {
            "com.example.some-label": "some-label-value",
            "image": "alpine",
            "name": "my-container"
        }
    },
    "time": 1461943101,
    "timeNano": 1461943101381709551
}
{
    "status": "attach",
    "id": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
    "from": "alpine",
    "Type": "container",
    "Action": "attach",
    "Actor": {
        "ID": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
        "Attributes": {
            "com.example.some-label": "some-label-value",
            "image": "alpine",
            "name": "my-container"
        }
    },
    "time": 1461943101,
    "timeNano": 1461943101383858412
}
{
    "Type": "network",
    "Action": "connect",
    "Actor": {
        "ID": "7dc8ac97d5d29ef6c31b6052f3938c1e8f2749abbd17d1bd1febf2608db1b474",
        "Attributes": {
            "container": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
            "name": "bridge",
            "type": "bridge"
        }
    },
    "time": 1461943101,
    "timeNano": 1461943101394865557
}
{
    "status": "start",
    "id": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
    "from": "alpine",
    "Type": "container",
    "Action": "start",
    "Actor": {
        "ID": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
        "Attributes": {
            "com.example.some-label": "some-label-value",
            "image": "alpine",
            "name": "my-container"
        }
    },
    "time": 1461943101,
    "timeNano": 1461943101607533796
}
{
    "status": "resize",
    "id": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
    "from": "alpine",
    "Type": "container",
    "Action": "resize",
    "Actor": {
        "ID": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
        "Attributes": {
            "com.example.some-label": "some-label-value",
            "height": "46",
            "image": "alpine",
            "name": "my-container",
            "width": "204"
        }
    },
    "time": 1461943101,
    "timeNano": 1461943101610269268
}
{
    "status": "die",
    "id": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
    "from": "alpine",
    "Type": "container",
    "Action": "die",
    "Actor": {
        "ID": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
        "Attributes": {
            "com.example.some-label": "some-label-value",
            "exitCode": "0",
            "image": "alpine",
            "name": "my-container"
        }
    },
    "time": 1461943105,
    "timeNano": 1461943105079144137
}
{
    "Type": "network",
    "Action": "disconnect",
    "Actor": {
        "ID": "7dc8ac97d5d29ef6c31b6052f3938c1e8f2749abbd17d1bd1febf2608db1b474",
        "Attributes": {
            "container": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
            "name": "bridge",
            "type": "bridge"
        }
    },
    "time": 1461943105,
    "timeNano": 1461943105230860245
}
{
    "status": "destroy",
    "id": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
    "from": "alpine",
    "Type": "container",
    "Action": "destroy",
    "Actor": {
        "ID": "ede54ee1afda366ab42f824e8a5ffd195155d853ceaec74a927f249ea270c743",
        "Attributes": {
```

```
        "com.example.some-label": "some-label-value",
        "image": "alpine",
        "name": "my-container"
      }
    },
    "time": 1461943105,
    "timeNano": 1461943105338056026
}
```

Query parameters:

- **since** – Timestamp. Show all events created since timestamp and then stream
- **until** – Timestamp. Show events created until given timestamp and stop streaming
- **filters** – A json encoded value of the filters (a map[string][]string) to process on the event list. Available filters:
- container=<string>; -- container to filter
- event=<string>; -- event to filter
- image=<string>; -- image to filter
- label=<string>; -- image and container label to filter
- type=<string>; -- either container or image or volume or network or daemon
- volume=<string>; -- volume to filter
- network=<string>; -- network to filter
- daemon=<string>; -- daemon name or id to filter

Status codes:

- **200** – no error
- **400** - bad parameter
- **500** – server error

GET A TARBALL CONTAINING ALL IMAGES IN A REPOSITORY

GET /images/(name)/get

Get a tarball containing all images and metadata for the repository specified by name.

If name is a specific name and tag (e.g. ubuntu:latest), then only that image (and its parents) are returned.

If name is an image ID, similarly only that image (and its parents) are returned, but with the exclusion of the 'repositories' file in the tarball, as there were no image names referenced.

See the image tarball format for more details.

Example request

```
GET /v1.24/images/ubuntu/get
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/x-tar

Binary data stream
```

Status codes:

- **200** – no error
- **500** – server error

GET A TARBALL CONTAINING ALL IMAGES

GET /images/get

Get a tarball containing all images and metadata for one or more repositories.

For each value of the names parameter: if it is a specific name and tag (e.g. ubuntu:latest), then only that image (and its parents) are returned; if it is an image ID, similarly only that image (and its parents) are returned and there would be no names referenced in the ‘repositories’ file for this image ID.

See the image tarball format for more details.

Example request

```
GET /v1.24/images/get?names=myname%2Fmyapp%3Alatest&names=busybox
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/x-tar

Binary data stream
```

Status codes:

- **200** – no error
- **500** – server error

LOAD A TARBALL WITH A SET OF IMAGES AND TAGS INTO DOCKER

POST /images/load

Load a set of images and tags into a Docker repository. See the image tarball format for more details.

Example request

```
POST /v1.24/images/load
Content-Type: application/x-tar
Content-Length: 12345

Tarball in body
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{"status":"Loading
layer","progressDetail":{"current":32768,"total":1292800},"progress":"[=
              ] 32.77 kB/1.293 MB","id":"8ac8bfaff55a"}
```

```
{"status":"Loading
layer","progressDetail":{"current":65536,"total":1292800},"progress":"[==
                ] 65.54 kB/1.293 MB","id":"8ac8bfaff55a"}
{"status":"Loading
layer","progressDetail":{"current":98304,"total":1292800},"progress":"[===
                ]    98.3 kB/1.293 MB","id":"8ac8bfaff55a"}
{"status":"Loading
layer","progressDetail":{"current":131072,"total":1292800},"progress":"[=====
                 ] 131.1 kB/1.293 MB","id":"8ac8bfaff55a"}
...
{"stream":"Loaded image: busybox:latest\n"}
```

Example response:

If the "quiet" query parameter is set to true / 1 (?quiet=1), progress details are suppressed, and only a confirmation message is returned once the action completes.

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{"stream":"Loaded image: busybox:latest\n"}
```

Query parameters:

- **quiet** – Boolean value, suppress progress details during load. Defaults to 0 / false if omitted.

Status codes:

- **200** – no error
- **500** – server error

IMAGE TARBALL FORMAT

An image tarball contains one directory per image layer (named using its long ID), each containing these files:

- VERSION: currently 1.0 - the file format version
- json: detailed layer information, similar to docker inspect layer_id
- layer.tar: A tarfile containing the filesystem changes in this layer

The layer.tar file contains aufs style .wh..wh.aufs files and directories for storing attribute changes and deletions.

If the tarball defines a repository, the tarball should also include a repositories file at the root that contains a list of repository and tag names mapped to layer IDs.

```
{"hello-world":
    {"latest": "565a9d68a73f6706862bfe8409a7f659776d4d60a8d096eb4a3cbce6999cc2a1"}
}
```

EXEC CREATE

POST /containers/(id or name)/exec

Sets up an exec instance in a running container id

Example request:

```
POST /v1.24/containers/e90e34656806/exec HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
  "AttachStdin": true,
  "AttachStdout": true,
  "AttachStderr": true,
  "Cmd": ["sh"],
  "DetachKeys": "ctrl-p,ctrl-q",
  "Privileged": true,
  "Tty": true,
  "User": "123:456"
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
      "Id": "f90e34656806",
      "Warnings":[]
}
```

JSON parameters:

- **AttachStdin** - Boolean value, attaches to stdin of the exec command.
- **AttachStdout** - Boolean value, attaches to stdout of the exec command.
- **AttachStderr** - Boolean value, attaches to stderr of the exec command.
- **DetachKeys** – Override the key sequence for detaching a container. Format is a single character [a-Z] or ctrl-<value> where <value> is one of: a-z, @, ^, [, , or _.
- **Tty** - Boolean value to allocate a pseudo-TTY.
- **Cmd** - Command to run specified as a string or an array of strings.
- **Privileged** - Boolean value, runs the exec process with extended privileges.
- **User** - A string value specifying the user, and optionally, group to run the exec process inside the container. Format is one of: "user", "user:group", "uid", or "uid:gid".

Status codes:

- **201** – no error
- **404** – no such container
- **409** - container is paused
- **500** - server error

EXEC START

POST /exec/(id)/start

Starts a previously set up exec instance id. If detach is true, this API returns after starting the execcommand. Otherwise, this API sets up an interactive session with the exec command.

Example request:

```
POST /v1.24/exec/e90e34656806/start HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
  "Detach": false,
  "Tty": false
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.docker.raw-stream


{{ STREAM }}
```

JSON parameters:

- **Detach -** Detach from the exec command.
- **Tty -** Boolean value to allocate a pseudo-TTY.

Status codes:

- **200** – no error
- **404** – no such exec instance
- **409** - container is paused

Stream details:

Similar to the stream behavior of POST /containers/(id or name)/attach API

EXEC RESIZE

POST /exec/(id)/resize

Resizes the tty session used by the exec command id. The unit is number of characters. This API is valid only if tty was specified as part of creating and starting the exec command.

Example request:

```
POST /v1.24/exec/e90e34656806/resize?h=40&w=80 HTTP/1.1
Content-Type: text/plain
```

Example response:

```
HTTP/1.1 201 Created
```

```
Content-Type: text/plain
```

Query parameters:

- **h** – height of tty session
- **w** – width

Status codes:

- **201** – no error
- **404** – no such exec instance

EXEC INSPECT

GET /exec/(id)/json

Return low-level information about the exec command id.

Example request:

```
GET /v1.24/exec/11fb006128e8ceb3942e7c58d77750f24210e35f879dd204ac975c184b820b39/json HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "CanRemove": false,
  "ContainerID": "b53ee82b53a40c7dca428523e34f741f3abc51d9f297a14ff874bf761b995126",
  "DetachKeys": "",
  "ExitCode": 2,
  "ID": "f33bbfb39f5b142420f4759b2348913bd4a8d1a6d7fd56499cb41a1bb91d7b3b",
  "OpenStderr": true,
  "OpenStdin": true,
  "OpenStdout": true,
  "ProcessConfig": {
    "arguments": [
      "-c",
      "exit 2"
    ],
    "entrypoint": "sh",
    "privileged": false,
    "tty": true,
    "user": "1000"
  },
  "Running": false
}
```

Status codes:

- **200** – no error
- **404** – no such exec instance

- **500** - server error

3.4 Volumes

LIST VOLUMES

`GET /volumes`

Example request:

```
GET /v1.24/volumes HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "Volumes": [
    {
      "Name": "tardis",
      "Driver": "local",
      "Mountpoint": "/var/lib/docker/volumes/tardis",
      "Labels": null,
      "Scope": "local"
    }
  ],
  "Warnings": []
}
```

Query parameters:

- **filters** - JSON encoded value of the filters (a `map[string][]string`) to process on the volumes list. Available filters:
  - `name=<volume-name>` Matches all or part of a volume name.
  - `dangling=<boolean>` When set to `true` (or `1`), returns all volumes that are "dangling" (not in use by a container). When set to `false` (or `0`), only volumes that are in use by one or more containers are returned.
  - `driver=<volume-driver-name>` Matches all or part of a volume driver name.

Status codes:

- **200** - no error
- **500** - server error

CREATE A VOLUME

`POST /volumes/create`

Create a volume

Example request:

```
POST /v1.24/volumes/create HTTP/1.1
Content-Type: application/json
```

```
Content-Length: 12345

{
  "Name": "tardis",
  "Labels": {
    "com.example.some-label": "some-value",
    "com.example.some-other-label": "some-other-value"
  },
  "Driver": "custom"
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "Name": "tardis",
  "Driver": "custom",
  "Mountpoint": "/var/lib/docker/volumes/tardis",
  "Status": {
    "hello": "world"
  },
  "Labels": {
    "com.example.some-label": "some-value",
    "com.example.some-other-label": "some-other-value"
  },
  "Scope": "local"
}
```

Status codes:

- **201** - no error
- **500** - server error

JSON parameters:

- **Name** - The new volume's name. If not specified, Docker generates a name.
- **Driver** - Name of the volume driver to use. Defaults to `local` for the name.
- **DriverOpts** - A mapping of driver options and values. These options are passed directly to the driver and are driver specific.
- **Labels** - Labels to set on the volume, specified as a map: `{"key":"value","key2":"value2"}`

JSON fields in response:

Refer to the inspect a volume section or details about the JSON fields returned in the response.

INSPECT A VOLUME

`GET /volumes/(name)`

Return low-level information on the volume `name`

Example request:

```
GET /v1.24/volumes/tardis
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "Name": "tardis",
  "Driver": "custom",
  "Mountpoint": "/var/lib/docker/volumes/tardis/_data",
  "Status": {
    "hello": "world"
  },
  "Labels": {
        "com.example.some-label": "some-value",
        "com.example.some-other-label": "some-other-value"
  },
  "Scope": "local"
}
```

Status codes:

- **200** - no error
- **404** - no such volume
- **500** - server error

JSON fields in response:

The following fields can be returned in the API response. Empty fields, or fields that are not supported by the volume's driver may be omitted in the response.

- **Name** - Name of the volume.
- **Driver** - Name of the volume driver used by the volume.
- **Mountpoint** - Mount path of the volume on the host.
- **Status** - Low-level details about the volume, provided by the volume driver. Details are returned as a map with key/value pairs: `{"key":"value","key2":"value2"}`. The `Status` field is optional, and is omitted if the volume driver does not support this feature.
- **Labels** - Labels set on the volume, specified as a map: `{"key":"value","key2":"value2"}`.
- **Scope** - Scope describes the level at which the volume exists, can be one of `global` for cluster-wide or `local` for machine level. The default is `local`.

REMOVE A VOLUME

`DELETE /volumes/(name)`

Instruct the driver to remove the volume (`name`).

Example request:

```
DELETE /v1.24/volumes/tardis HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Status codes:

- **204** - no error
- **404** - no such volume or volume driver
- **409** - volume is in use and cannot be removed
- **500** - server error

3.5 Networks

LIST NETWORKS

GET /networks

Example request:

```
GET /v1.24/networks?filters={"type":{"custom":true}} HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "Name": "bridge",
    "Id": "f2de39df4171b0dc801e8002d1d999b77256983dfc63041c0f34030aa3977566",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "Internal": false,
    "IPAM": {
      "Driver": "default",
      "Config": [
        {
          "Subnet": "172.17.0.0/16"
        }
      ]
    },
    "Containers": {
      "39b69226f9d79f5634485fb236a23b2fe4e96a0a94128390a7fbbcc167065867": {
        "EndpointID": "ed2419a97c1d9954d05b46e462e7002ea552f216e9b136b80a7db8d98b442eda",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
            "IPv6Address": ""
          }
        },
        "Options": {
          "com.docker.network.bridge.default_bridge": "true",
          "com.docker.network.bridge.enable_icc": "true",
          "com.docker.network.bridge.enable_ip_masquerade": "true",
          "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
          "com.docker.network.bridge.name": "docker0",
          "com.docker.network.driver.mtu": "1500"
        }
      },
      {
        "Name": "none",
        "Id": "e086a3893b05ab69242d3c44e49483a3bbbd3a26b46baa8f61ab797c1088d794",
        "Scope": "local",
        "Driver": "null",
        "EnableIPv6": false,
        "Internal": false,
        "IPAM": {
          "Driver": "default",
          "Config": []
        },
        "Containers": {},
        "Options": {}
      },
      {
        "Name": "host",
        "Id": "13e871235c677f196c4e1ecebb9dc733b9b2d2ab589e30c539efeda84a24215e",
        "Scope": "local",
        "Driver": "host",
        "EnableIPv6": false,
        "Internal": false,
        "IPAM": {
          "Driver": "default",
          "Config": []
        },
        "Containers": {},
        "Options": {}
      }
    ]
```

Query parameters:

- **filters** - JSON encoded network list filter. The filter value is one of:

- driver=<driver-name> Matches a network's driver.
- id=<network-id> Matches all or part of a network id.
- label=<key> or label=<key>=<value> of a network label.
- name=<network-name> Matches all or part of a network name.
- type=["custom"|"builtin"] Filters networks by type. The custom keyword returns all user-defined networks.

Status codes:

- **200** - no error
- **500** - server error

INSPECT NETWORK

GET /networks/(id or name)

Return low-level information on the network id

Example request:

```
GET /v1.24/networks/7d86d31b1478e7cca9ebed7e73aa0fdeec46c5ca29497431d3007d2d9e15ed99 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "Name": "net01",
  "Id": "7d86d31b1478e7cca9ebed7e73aa0fdeec46c5ca29497431d3007d2d9e15ed99",
  "Scope": "local",
  "Driver": "bridge",
  "EnableIPv6": false,
  "IPAM": {
    "Driver": "default",
    "Config": [
      {
        "Subnet": "172.19.0.0/16",
        "Gateway": "172.19.0.1"
      }
    ],
    "Options": {
        "foo": "bar"
    }
  },
  "Internal": false,
  "Containers": {
    "19a4d5d687db25203351ed79d478946f861258f018fe384f229f2efa4b23513c": {
      "Name": "test",
      "EndpointID": "628cadb8bcb92de107b2a1e516cbffe463e321f548feb37697cce00ad694f21a",
          "MacAddress": "02:42:ac:13:00:02",
          "IPv4Address": "172.19.0.2/16",
          "IPv6Address": ""
        }
    },
    "Options": {
        "com.docker.network.bridge.default_bridge": "true",
        "com.docker.network.bridge.enable_icc": "true",
        "com.docker.network.bridge.enable_ip_masquerade": "true",
        "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
        "com.docker.network.bridge.name": "docker0",
        "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {
        "com.example.some-label": "some-value",
        "com.example.some-other-label": "some-other-value"
    }
}
```

Status codes:

- **200** - no error
- **404** - network not found
- **500** - server error

CREATE A NETWORK

POST /networks/create

Create a network

Example request:

```
POST /v1.24/networks/create HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
    "Name":"isolated_nw",
    "CheckDuplicate":true,
    "Driver":"bridge",
    "EnableIPv6": true,
    "IPAM":{
        "Driver": "default",
        "Config":[
          {
            "Subnet":"172.20.0.0/16",
            "IPRange":"172.20.10.0/24",
            "Gateway":"172.20.10.11"
          },
          {
            "Subnet":"2001:db8:abcd::/64",
            "Gateway":"2001:db8:abcd::1011"
          }
        ],
        "Options": {
          "foo": "bar"
        }
    },
    "Internal":true,
    "Options": {
        "com.docker.network.bridge.default_bridge": "true",
        "com.docker.network.bridge.enable_icc": "true",
        "com.docker.network.bridge.enable_ip_masquerade": "true",
        "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
        "com.docker.network.bridge.name": "docker0",
        "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {
        "com.example.some-label": "some-value",
        "com.example.some-other-label": "some-other-value"
    }
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "Id": "22be93d5babb089c5aab8dbc369042fad48ff791584ca2da2100db837a1c7c30",
    "Warning": ""
}
```

Status codes:

- **201 -** no error
- **403 -** operation not supported for pre-defined networks
- **404 -** plugin not found
- **500 -** server error

JSON parameters:

- **Name -** The new network’s name. this is a mandatory field
- **CheckDuplicate -** Requests daemon to check for networks with same name. Defaults to `false`. Since Network is primarily keyed based on a random ID and not on the name, and network name is

strictly a user-friendly alias to the network which is uniquely identified using ID, there is no guaranteed way to check for duplicates. This parameter CheckDuplicate is there to provide a best effort checking of any networks which has the same name but it is not guaranteed to catch all name collisions.

- **Driver -** Name of the network driver plugin to use. Defaults to bridge driver
- **Internal -** Restrict external access to the network
- **IPAM -** Optional custom IP scheme for the network
  - **Driver -** Name of the IPAM driver to use. Defaults to default driver
  - **Config -** List of IPAM configuration options, specified as a map:{"Subnet": <CIDR>, "IPRange": <CIDR>, "Gateway": <IP address>, "AuxAddress": <device_name:IP address>}
  - **Options -** Driver-specific options, specified as a map: {"option":"value" [,"option2":"value2"]}
- **EnableIPv6 -** Enable IPv6 on the network
- **Options -** Network specific options to be used by the drivers
- **Labels -** Labels to set on the network, specified as a map: {"key":"value" [,"key2":"value2"]}

CONNECT A CONTAINER TO A NETWORK

POST /networks/(id or name)/connect

Connect a container to a network

Example request:

```
POST /v1.24/networks/22be93d5babb089c5aab8dbc369042fad48ff791584ca2da2100db837a1c7c30/connect
HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
  "Container":"3613f73ba0e4",
  "EndpointConfig": {
    "IPAMConfig": {
        "IPv4Address":"172.24.56.89",
        "IPv6Address":"2001:db8::5689"
    }
  }
}
```

Example response:

```
HTTP/1.1 200 OK
```

Status codes:

- **200 -** no error
- **403 -** operation not supported for swarm scoped networks
- **404 -** network or container is not found
- **500 -** Internal Server Error

JSON parameters:

- **container -** container-id/name to be connected to the network

DISCONNECT A CONTAINER FROM A NETWORK

POST /networks/(id or name)/disconnect

Disconnect a container from a network

Example request:

```
POST /v1.24/networks/22be93d5babb089c5aab8dbc369042fad48ff791584ca2da2100db837a1c7c30/disconnect
HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
  "Container":"3613f73ba0e4",
  "Force":false
}
```

Example response:

```
HTTP/1.1 200 OK
```

Status codes:

- **200** - no error
- **403** - operation not supported for swarm scoped networks
- **404** - network or container not found
- **500** - Internal Server Error

JSON parameters:

- **Container** - container-id/name to be disconnected from a network
- **Force** - Force the container to disconnect from a network

REMOVE A NETWORK

DELETE /networks/(id or name)

Instruct the driver to remove the network (id).

Example request:

```
DELETE /v1.24/networks/22be93d5babb089c5aab8dbc369042fad48ff791584ca2da2100db837a1c7c30 HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

Status codes:

- **204** - no error
- **403** - operation not supported for pre-defined networks
- **404** - no such network
- **500** - server error

3.6 Plugins (experimental)

LIST PLUGINS

GET /plugins

Returns information about installed plugins.

Example request:

```
GET /v1.24/plugins HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "Id": "5724e2c8652da337ab2eedd19fc6fc0ec908e4bd907c7421bf6a8dfc70c4c078",
    "Name": "tiborvass/no-remove",
    "Tag": "latest",
    "Active": true,
    "Config": {
      "Mounts": [
        {
          "Name": "",
          "Description": "",
          "Settable": null,
          "Source": "/data",
          "Destination": "/data",
          "Type": "bind",
          "Options": [
            "shared",
            "rbind"
          ]
        },
        {
          "Name": "",
          "Description": "",
          "Settable": null,
          "Source": null,
          "Destination": "/foobar",
          "Type": "tmpfs",
          "Options": null
        }
      ],
      "Env": [
        "DEBUG=1"
      ],
    "Args": null,
    "Devices": null
  },
  "Manifest": {
    "ManifestVersion": "v0",
    "Description": "A test plugin for Docker",
    "Documentation": "https://docs.docker.com/engine/extend/plugins/",
    "Interface": {
      "Types": [
        "docker.volumedriver/1.0"
      ],
      "Socket": "plugins.sock"
    },
    "Entrypoint": [
      "plugin-no-remove",
      "/data"
    ],
    "Workdir": "",
    "User": {
    },
    "Network": {
      "Type": "host"
    },
    "Capabilities": null,
    "Mounts": [
      {
        "Name": "",
        "Description": "",
        "Settable": null,
        "Source": "/data",
        "Destination": "/data",
        "Type": "bind",
        "Options": [
          "shared",
          "rbind"
        ]
      },
      {
        "Name": "",
        "Description": "",
        "Settable": null,
        "Source": null,
        "Destination": "/foobar",
            "Type": "tmpfs",
            "Options": null
          }
        ],
        "Devices": [
          {
            "Name": "device",
            "Description": "a host device to mount",
            "Settable": null,
            "Path": "/dev/cpu_dma_latency"
          }
        ],
        "Env": [
          {
            "Name": "DEBUG",
            "Description": "If set, prints debug messages",
            "Settable": null,
            "Value": "1"
          }
        ],
        "Args": {
          "Name": "args",
          "Description": "command line arguments",
          "Settable": null,
          "Value": [

          ]
        }
      }
    }
]
```

Status codes:

- **200** - no error
- **500** - server error

INSTALL A PLUGIN

POST /plugins/pull?name=<plugin name>

Pulls and installs a plugin. After the plugin is installed, it can be enabled using the POST /plugins/(plugin name)/enable endpoint.

Example request:

```
POST /v1.24/plugins/pull?name=tiborvass/no-remove:latest HTTP/1.1
```

The :latest tag is optional, and is used as default if omitted. When using this endpoint to pull a plugin from the registry, the X-Registry-Auth header can be used to include a base64-encoded AuthConfig object. Refer to the create an image section for more details.

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 175

[
  {
    "Name": "network",
    "Description": "",
    "Value": [
      "host"
    ]
  },
  {
    "Name": "mount",
    "Description": "",
    "Value": [
      "/data"
    ]
  },
  {
    "Name": "device",
    "Description": "",
    "Value": [
      "/dev/cpu_dma_latency"
    ]
  }
]
```

Query parameters:

- **name** - Name of the plugin to pull. The name may include a tag or digest. This parameter is required.

Status codes:

- **200** - no error
- **500** - error parsing reference / not a valid repository/tag: repository name must have at least one component
- **500** - plugin already exists

INSPECT A PLUGIN

GET /plugins/(plugin name)

Returns detailed information about an installed plugin.

Example request:

```
GET /v1.24/plugins/tiborvass/no-remove:latest HTTP/1.1
```

The `:latest` tag is optional, and is used as default if omitted.

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "Id": "5724e2c8652da337ab2eedd19fc6fc0ec908e4bd907c7421bf6a8dfc70c4c078",
  "Name": "tiborvass/no-remove",
  "Tag": "latest",
  "Active": false,
  "Config": {
    "Mounts": [
      {
        "Name": "",
        "Description": "",
        "Settable": null,
        "Source": "/data",
        "Destination": "/data",
        "Type": "bind",
        "Options": [
          "shared",
          "rbind"
        ]
      },
      {
        "Name": "",
        "Description": "",
        "Settable": null,
        "Source": null,
        "Destination": "/foobar",
        "Type": "tmpfs",
        "Options": null
      }
    ],
    "Env": [
      "DEBUG=1"
    ],
    "Args": null,
    "Devices": null
```

```
},
"Manifest": {
    "ManifestVersion": "v0",
    "Description": "A test plugin for Docker",
    "Documentation": "https://docs.docker.com/engine/extend/plugins/",
    "Interface": {
        "Types": [
            "docker.volumedriver/1.0"
        ],
        "Socket": "plugins.sock"
    },
    "Entrypoint": [
        "plugin-no-remove",
        "/data"
    ],
    "Workdir": "",
    "User": {
    },
    "Network": {
        "Type": "host"
    },
    "Capabilities": null,
    "Mounts": [
        {
            "Name": "",
            "Description": "",
            "Settable": null,
            "Source": "/data",
            "Destination": "/data",
            "Type": "bind",
            "Options": [
                "shared",
                "rbind"
            ]
        },
        {
            "Name": "",
            "Description": "",
            "Settable": null,
            "Source": null,
            "Destination": "/foobar",
            "Type": "tmpfs",
            "Options": null
        }
      ],
      "Devices": [
        {
          "Name": "device",
          "Description": "a host device to mount",
          "Settable": null,
          "Path": "/dev/cpu_dma_latency"
        }
      ],
      "Env": [
        {
          "Name": "DEBUG",
          "Description": "If set, prints debug messages",
          "Settable": null,
          "Value": "1"
        }
      ],
      "Args": {
        "Name": "args",
        "Description": "command line arguments",
        "Settable": null,
        "Value": [

        ]
      }
  }
}
```

Status codes:

- **200 -** no error
- **404 -** plugin not installed

ENABLE A PLUGIN

POST /plugins/(plugin name)/enable

Enables a plugin

Example request:

```
POST /v1.24/plugins/tiborvass/no-remove:latest/enable HTTP/1.1
```

The :latest tag is optional, and is used as default if omitted.

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
```

```
Content-Type: text/plain; charset=utf-8
```

Status codes:

- **200** - no error
- **404** - plugin not installed
- **500** - plugin is already enabled

DISABLE A PLUGIN

POST /plugins/(plugin name)/disable

Disables a plugin

Example request:

```
POST /v1.24/plugins/tiborvass/no-remove:latest/disable HTTP/1.1
```

The :latest tag is optional, and is used as default if omitted.

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

Status codes:

- **200** - no error
- **404** - plugin not installed
- **500** - plugin is already disabled

REMOVE A PLUGIN

DELETE /plugins/(plugin name)

Removes a plugin

Example request:

```
DELETE /v1.24/plugins/tiborvass/no-remove:latest HTTP/1.1
```

The :latest tag is optional, and is used as default if omitted.

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

Status codes:

- **200** - no error
- **404** - plugin not installed
- **500** - plugin is active

3.7 Nodes

Note: Node operations require the engine to be part of a swarm.

LIST NODES

GET /nodes

List nodes

Example request:

```
GET /v1.24/nodes HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "ID": "24ifsmvkjbyhk",
    "Version": {
      "Index": 8
    },
    "CreatedAt": "2016-06-07T20:31:11.853781916Z",
    "UpdatedAt": "2016-06-07T20:31:11.999868824Z",
    "Spec": {
      "Name": "my-node",
      "Role": "manager",
      "Availability": "active"
      "Labels": {
          "foo": "bar"
      }
    },
    "Description": {
      "Hostname": "bf3067039e47",
      "Platform": {
        "Architecture": "x86_64",
        "OS": "linux"
      },
      "Resources": {
        "NanoCPUs": 4000000000,
        "MemoryBytes": 8272408576
      },
      "Engine": {
        "EngineVersion": "1.12.0",
        "Labels": {
            "foo": "bar",
        }
        "Plugins": [
          {
                "Type": "Volume",
                "Name": "local"
              },
              {
                "Type": "Network",
                "Name": "bridge"
              }
              {
                "Type": "Network",
                "Name": "null"
              }
              {
                "Type": "Network",
                "Name": "overlay"
              }
            ]
          }
        },
        "Status": {
          "State": "ready"
        },
        "ManagerStatus": {
          "Leader": true,
          "Reachability": "reachable",
          "Addr": "172.17.0.2:2377""
        }
      }
]
```

Query parameters:

- **filters** – a JSON encoded value of the filters (a map[string][]string) to process on the nodes list. Available filters:
    - id=<node id>
    - label=<engine label>
    -

      membership=(accepted pending)`
    -
    - name=<node name>
    -

      role=(manager worker)`
    -

Status codes:

- **200** – no error
- **406** - node is not part of a swarm
- **500** – server error

INSPECT A NODE

GET /nodes/(id or name)

Return low-level information on the node id

Example request:

```
GET /v1.24/nodes/24ifsmvkjbyhk HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "ID": "24ifsmvkjbyhk",
  "Version": {
    "Index": 8
  },
  "CreatedAt": "2016-06-07T20:31:11.853781916Z",
  "UpdatedAt": "2016-06-07T20:31:11.999868824Z",
  "Spec": {
    "Name": "my-node",
    "Role": "manager",
    "Availability": "active"
    "Labels": {
        "foo": "bar"
    }
  },
  "Description": {
    "Hostname": "bf3067039e47",
    "Platform": {
      "Architecture": "x86_64",
      "OS": "linux"
    },
    "Resources": {
      "NanoCPUs": 4000000000,
      "MemoryBytes": 8272408576
    },
    "Engine": {
      "EngineVersion": "1.12.0",
      "Labels": {
          "foo": "bar",
      }
```

```
            "Plugins": [
              {
                "Type": "Volume",
                "Name": "local"
              },
              {
                "Type": "Network",
                "Name": "bridge"
              }
              {
                "Type": "Network",
                "Name": "null"
              }
              {
                "Type": "Network",
                "Name": "overlay"
              }
            ]
          }
        },
        "Status": {
          "State": "ready"
        },
        "ManagerStatus": {
          "Leader": true,
          "Reachability": "reachable",
          "Addr": "172.17.0.2:2377""
        }
    }
```

Status codes:

- **200** – no error
- **404** – no such node
- **406** – node is not part of a swarm
- **500** – server error

REMOVE A NODE

DELETE /nodes/(id or name)

Remove a node from the swarm.

Example request:

```
DELETE /v1.24/nodes/24ifsmvkjbyhk HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

Query parameters:

- **force** - 1/True/true or 0/False/false, Force remove a node from the swarm. Default false.

Status codes:

- **200** – no error
- **404** – no such node
- **406** – node is not part of a swarm
- **500** – server error

UPDATE A NODE

POST /nodes/(id)/update

Update a node.

The payload of the POST request is the new NodeSpec and overrides the current NodeSpec for the specified node.

If Availability or Role are omitted, this returns an error. Any other field omitted resets the current value to either an empty value or the default cluster-wide value.

Example Request

```
POST /v1.24/nodes/24ifsmvkjbyhk/update?version=8 HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
  "Availability": "active",
  "Name": "node-name",
  "Role": "manager",
  "Labels": {
    "foo": "bar"
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

Query parameters:

- **version** – The version number of the node object being updated. This is required to avoid conflicting writes.

JSON Parameters:

- **Annotations** – Optional medata to associate with the node.

- **Name** – User-defined name for the node.
- **Labels** – A map of labels to associate with the node (e.g.,{"key":"value", "key2":"value2"}).
-

Role - Role of the node (worker manager).

-
-

Availability - Availability of the node (active pause drain).

-

Status codes:

- **200** – no error
- **404** – no such node
- **406** – node is not part of a swarm
- **500** – server error

3.8 Swarm

INSPECT SWARM

GET /swarm

Inspect swarm

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "CreatedAt" : "2016-08-15T16:00:20.349727406Z",
  "Spec" : {
    "Dispatcher" : {
      "HeartbeatPeriod" : 5000000000
    },
    "Orchestration" : {
     "TaskHistoryRetentionLimit" : 10
    },
    "CAConfig" : {
      "NodeCertExpiry" : 7776000000000000
    },
    "Raft" : {
      "LogEntriesForSlowFollowers" : 500,
      "HeartbeatTick" : 1,
      "SnapshotInterval" : 10000,
      "ElectionTick" : 3
    },
    "TaskDefaults" : {},
        "Name" : "default"
    },
  "JoinTokens" : {
        "Worker" :
"SWMTKN-1-1h8aps2yszaiqmz2l3oc5392pgk8e49qhx2aj3nyv0ui0hez2a-6qmn92w6bu3jdvnglku58u11a",
        "Manager" :
"SWMTKN-1-1h8aps2yszaiqmz2l3oc5392pgk8e49qhx2aj3nyv0ui0hez2a-8llk83c4wm9lwioey2s316r9l"
  },
  "ID" : "70ilmkj2f6sp2137c753w2nmt",
  "UpdatedAt" : "2016-08-15T16:32:09.623207604Z",
  "Version" : {
      "Index" : 51
}     }
```

Status codes:

- **200** - no error
- **406** – node is not part of a swarm
- **500** - sever error

INITIALIZE A NEW SWARM

POST /swarm/init

Initialize a new swarm. The body of the HTTP response includes the node ID.

Example request:

```
POST /v1.24/swarm/init HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
    "ListenAddr": "0.0.0.0:2377",
    "AdvertiseAddr": "192.168.1.1:2377",
    "ForceNewCluster": false,
    "Spec": {
        "Orchestration": {},
        "Raft": {},
        "Dispatcher": {},
        "CAConfig": {}
    }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 28
Content-Type: application/json
```

```
Date: Thu, 01 Sep 2016 21:49:13 GMT
Server: Docker/1.12.0 (linux)

"7v2t30z9blmxuhnyo6s4cpenp"
```

Status codes:

- **200** – no error
- **400** – bad parameter
- **406** – node is already part of a swarm
- **500** - server error

JSON Parameters:

- **ListenAddr** – Listen address used for inter-manager communication, as well as determining the networking interface used for the VXLAN Tunnel Endpoint (VTEP). This can either be an address/port combination in the form `192.168.1.1:4567`, or an interface followed by a port number, like `eth0:4567`. If the port number is omitted, the default swarm listening port is used.
- **AdvertiseAddr** – Externally reachable address advertised to other nodes. This can either be an address/port combination in the form `192.168.1.1:4567`, or an interface followed by a port number, like `eth0:4567`. If the port number is omitted, the port number from the listen address is used. If `AdvertiseAddr` is not specified, it will be automatically detected when possible.
- **ForceNewCluster** – Force creation of a new swarm.
- **Spec** – Configuration settings for the new swarm.
  - **Orchestration** – Configuration settings for the orchestration aspects of the swarm.
    - **TaskHistoryRetentionLimit** – Maximum number of tasks history stored.
  - **Raft** – Raft related configuration.
    - **SnapshotInterval** – Number of logs entries between snapshot.
    - **KeepOldSnapshots** – Number of snapshots to keep beyond the current snapshot.
    - **LogEntriesForSlowFollowers** – Number of log entries to keep around to sync up slow followers after a snapshot is created.
    - **HeartbeatTick** – Amount of ticks (in seconds) between each heartbeat.
    - **ElectionTick** – Amount of ticks (in seconds) needed without a leader to trigger a new election.
  - **Dispatcher** – Configuration settings for the task dispatcher.
    - **HeartbeatPeriod** – The delay for an agent to send a heartbeat to the dispatcher.
  - **CAConfig** – Certificate authority configuration.
    - **NodeCertExpiry** – Automatic expiry for nodes certificates.
    - **ExternalCA** - Configuration for forwarding signing requests to an external certificate authority.
      - **Protocol** - Protocol for communication with the external CA (currently only “cfssl” is supported).
      - **URL** - URL where certificate signing requests should be sent.
      - **Options** - An object with key/value pairs that are interpreted as protocol-specific options for the external CA driver.

JOIN AN EXISTING SWARM

POST /swarm/join

Join an existing swarm

Example request:

```
POST /v1.24/swarm/join HTTP/1.1
Content-Type: application/json

{
  "ListenAddr": "0.0.0.0:2377",
  "AdvertiseAddr": "192.168.1.1:2377",
  "RemoteAddrs": ["node1:2377"],
  "JoinToken": "SWMTKN-1-3pu6hszjas19xyp7ghgosyx9k8atbfcr8p2is99znpy26u2lkl-7p73s1dx5in4tatdymyhg9hu2"
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

Status codes:

- **200** – no error
- **400** – bad parameter
- **406** – node is already part of a swarm
- **500** - server error

JSON Parameters:

- **ListenAddr** – Listen address used for inter-manager communication if the node gets promoted to manager, as well as determining the networking interface used for the VXLAN Tunnel Endpoint (VTEP).
- **AdvertiseAddr** – Externally reachable address advertised to other nodes. This can either be an address/port combination in the form 192.168.1.1:4567, or an interface followed by a port number, like eth0:4567. If the port number is omitted, the port number from the listen address is used. If AdvertiseAddr is not specified, it will be automatically detected when possible.
- **RemoteAddr** – Address of any manager node already participating in the swarm.
- **JoinToken** – Secret token for joining this swarm.

LEAVE A SWARM

POST /swarm/leave

Leave a swarm

Example request:

```
POST /v1.24/swarm/leave HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
```

```
Content-Type: text/plain; charset=utf-8
```

Query parameters:

- **force** - Boolean (0/1, false/true). Force leave swarm, even if this is the last manager or that it will break the cluster.

Status codes:

- **200** – no error
- **406** – node is not part of a swarm
- **500** - server error

UPDATE A SWARM

POST /swarm/update

Update a swarm

Example request:

```
POST /v1.24/swarm/update HTTP/1.1
Content-Length: 12345

{
  "Name": "default",
  "Orchestration": {
    "TaskHistoryRetentionLimit": 10
  },
  "Raft": {
    "SnapshotInterval": 10000,
    "LogEntriesForSlowFollowers": 500,
    "HeartbeatTick": 1,
    "ElectionTick": 3
  },
  "Dispatcher": {
    "HeartbeatPeriod": 5000000000
  },
  "CAConfig": {
    "NodeCertExpiry": 7776000000000000
  },
  "JoinTokens": {
    "Worker":
"SWMTKN-1-3pu6hszjas19xyp7ghgosyx9k8atbfcr8p2is99znpy26u2lkl-1awxwuwd3z9j1z3puu7rcgdbx",
    "Manager":
"SWMTKN-1-3pu6hszjas19xyp7ghgosyx9k8atbfcr8p2is99znpy26u2lkl-7p73s1dx5in4tatdymyhg9hu2"
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

Query parameters:

- **version** – The version number of the swarm object being updated. This is required to avoid conflicting writes.
- **rotateWorkerToken** - Set to `true` (or `1`) to rotate the worker join token.
- **rotateManagerToken** - Set to `true` (or `1`) to rotate the manager join token.

Status codes:

- **200** – no error
- **400** – bad parameter
- **406** – node is not part of a swarm
- **500** - server error

JSON Parameters:

- **Orchestration** – Configuration settings for the orchestration aspects of the swarm.
  - **TaskHistoryRetentionLimit** – Maximum number of tasks history stored.
- **Raft** – Raft related configuration.
  - **SnapshotInterval** – Number of logs entries between snapshot.
  - **KeepOldSnapshots** – Number of snapshots to keep beyond the current snapshot.
  - **LogEntriesForSlowFollowers** – Number of log entries to keep around to sync up slow followers after a snapshot is created.
  - **HeartbeatTick** – Amount of ticks (in seconds) between each heartbeat.
  - **ElectionTick** – Amount of ticks (in seconds) needed without a leader to trigger a new election.
- **Dispatcher** – Configuration settings for the task dispatcher.
  - **HeartbeatPeriod** – The delay for an agent to send a heartbeat to the dispatcher.
- **CAConfig** – CA configuration.
  - **NodeCertExpiry** – Automatic expiry for nodes certificates.
  - **ExternalCA** - Configuration for forwarding signing requests to an external certificate authority.
    - **Protocol** - Protocol for communication with the external CA (currently only "cfssl" is supported).
    - **URL** - URL where certificate signing requests should be sent.
    - **Options** - An object with key/value pairs that are interpreted as protocol-specific options for the external CA driver.
- **JoinTokens** - Tokens that can be used by other nodes to join the swarm.
  - **Worker** - Token to use for joining as a worker.
  - **Manager** - Token to use for joining as a manager.

3.9 Services

Note: Service operations require to first be part of a swarm.

LIST SERVICES

GET /services

List services

Example request:

```
GET /v1.24/services HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "ID": "9mnpnzenvg8p8tdbtq4wvbkcz",
    "Version": {
      "Index": 19
    },
    "CreatedAt": "2016-06-07T21:05:51.880065305Z",
    "UpdatedAt": "2016-06-07T21:07:29.962229872Z",
    "Spec": {
      "Name": "hopeful_cori",
      "TaskTemplate": {
        "ContainerSpec": {
          "Image": "redis"
        },
        "Resources": {
          "Limits": {},
          "Reservations": {}
        },
        "RestartPolicy": {
          "Condition": "any",
          "MaxAttempts": 0
        },
        "Placement": {
          "Constraints": [
            "node.role == worker"
          ]
        }
      },
      "Mode": {
        "Replicated": {
          "Replicas": 1
        }
      },
      "UpdateConfig": {
            "Parallelism": 1,
            "FailureAction": "pause"
        },
        "EndpointSpec": {
            "Mode": "vip",
            "Ports": [
                {
                    "Protocol": "tcp",
                    "TargetPort": 6379,
                    "PublishedPort": 30001
                }
            ]
        }
    },
    "Endpoint": {
        "Spec": {
            "Mode": "vip",
            "Ports": [
                {
                    "Protocol": "tcp",
                    "TargetPort": 6379,
                    "PublishedPort": 30001
                }
            ]
        },
        "Ports": [
            {
                "Protocol": "tcp",
                "TargetPort": 6379,
                "PublishedPort": 30001
            }
        ],
        "VirtualIPs": [
            {
                "NetworkID": "4qvuz4ko70xaltuqbt8956gd1",
                "Addr": "10.255.0.2/16"
            },
            {
                "NetworkID": "4qvuz4ko70xaltuqbt8956gd1",
                "Addr": "10.255.0.3/16"
            }
        ]
    }
```

```
  }
]
```

Query parameters:

- **filters** – a JSON encoded value of the filters (a map[string][]string) to process on the services list. Available filters:
  - id=<service id>
  - label=<service label>
  - name=<service name>

Status codes:

- **200** – no error
- **406** – node is not part of a swarm
- **500** – server error

CREATE A SERVICE

POST /services/create

Create a service. When using this endpoint to create a service using a private repository from the registry, the X-Registry-Auth header must be used to include a base64-encoded AuthConfig object. Refer to the create an image section for more details.

Example request:

```
POST /v1.24/services/create HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
  "Name": "web",
  "TaskTemplate": {
    "ContainerSpec": {
      "Image": "nginx:alpine",
      "Mounts": [
        {
          "ReadOnly": true,
          "Source": "web-data",
          "Target": "/usr/share/nginx/html",
          "Type": "volume",
          "VolumeOptions": {
            "DriverConfig": {
            },
            "Labels": {
              "com.example.something": "something-value"
            }
          }
        }
        ],
        "User": "33"
    },
    "Networks": [
            {
                "Target": "overlay1"
            }
    ],
    "LogDriver": {
        "Name": "json-file",
        "Options": {
            "max-file": "3",
            "max-size": "10M"
        }
    },
    "Placement": {
        "Constraints": [
            "node.role == worker"
        ]
    },
    "Resources": {
        "Limits": {
            "MemoryBytes": 104857600
        },
        "Reservations": {
        }
    },
    "RestartPolicy": {
        "Condition": "on-failure",
        "Delay": 10000000000,
        "MaxAttempts": 10
    }
},
"Mode": {
    "Replicated": {
        "Replicas": 4
    }
},
"UpdateConfig": {
    "Delay": 30000000000,
    "Parallelism": 2,
    "FailureAction": "pause"
},
  "EndpointSpec": {
    "Ports": [
      {
        "Protocol": "tcp",
        "PublishedPort": 8080,
        "TargetPort": 80
      }
    ]
  },
  "Labels": {
    "foo": "bar"
  }
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "ID":"ak7w3gjqoa3kuz8xcpnyy0pvl"
}
```

Status codes:

- **201** – no error
- **403** - network is not eligible for services
- **406** – node is not part of a swarm
- **409** – name conflicts with an existing object
- **500** - server error

JSON Parameters:

- **Name** – User-defined name for the service.
- **Labels** – A map of labels to associate with the service (e.g., {"key":"value", "key2":"value2"}).
- **TaskTemplate** – Specification of the tasks to start as part of the new service.
    - **ContainerSpec** - Container settings for containers started as part of this task.
        - **Image** – A string specifying the image name to use for the container.
        - **Command** – The command to be run in the image.
        - **Args** – Arguments to the command.
        - **Env** – A list of environment variables in the form of ["VAR=value"[,"VAR2=value2"]].
        - **Dir** – A string specifying the working directory for commands to run in.
        - **User** – A string value specifying the user inside the container.
        - **Labels** – A map of labels to associate with the service (e.g.,{"key":"value", "key2":"value2"}).
        - **Mounts** – Specification for mounts to be added to containers created as part of the service.

            - **Target** – Container path.
            - **Source** – Mount source (e.g. a volume name, a host path).
            - **Type** – The mount type (bind, or volume).
            - **ReadOnly** – A boolean indicating whether the mount should be read-only.
            - **BindOptions** - Optional configuration for the bind type.
                - **Propagation** – A propagation mode with the value [r]private, [r]shared, or [r]slave.
            - **VolumeOptions** – Optional configuration for the volume type.
                - **NoCopy** – A boolean indicating if volume should be populated with the data from the target. (Default false)
                - **Labels** – User-defined name and labels for the volume.
                - **DriverConfig** – Map of driver-specific options.
                    - **Name** - Name of the driver to use to create the volume.
                    - **Options** - key/value map of driver specific options.
        - **StopGracePeriod** – Amount of time to wait for the container to terminate before forcefully killing it.
    - **LogDriver** - Log configuration for containers created as part of the service.
        - **Name** - Name of the logging driver to use (json-file, syslog, journald, gelf, fluentd, awslogs, splunk, etwlogs, none).
        - **Options** - Driver-specific options.
    - **Resources** – Resource requirements which apply to each individual container created as part of the service.
        - **Limits** – Define resources limits.
            - **NanoCPUs** – CPU limit in units of 10-9 CPU shares.
            - **MemoryBytes** – Memory limit in Bytes.
        - **Reservation** – Define resources reservation.
            - **NanoCPUs** – CPU reservation in units of 10-9 CPU shares.
            - **MemoryBytes** – Memory reservation in Bytes.
    - **RestartPolicy** – Specification for the restart policy which applies to containers created as part of this service.
        - **Condition** – Condition for restart (none, on-failure, or any).
        - **Delay** – Delay between restart attempts.
        - **MaxAttempts** – Maximum attempts to restart a given container before giving up (default value is 0, which is ignored).
        - **Window** – Windows is the time window used to evaluate the restart policy (default value is 0, which is unbounded).
    - **Placement** – Restrictions on where a service can run.
        - **Constraints** – An array of constraints, e.g. ["node.role == manager"].
- **Mode** – Scheduling mode for the service (replicated or global, defaults to replicated).
- **UpdateConfig** – Specification for the update strategy of the service.
    - **Parallelism** – Maximum number of tasks to be updated in one iteration (0 means unlimited parallelism).
    - **Delay** – Amount of time between updates.

  - **FailureAction** - Action to take if an updated task fails to run, or stops running during the update. Values are continue and pause.
- **Networks** – Array of network names or IDs to attach the service to.
- **EndpointSpec** – Properties that can be configured to access and load balance a service.
  - **Mode** – The mode of resolution to use for internal load balancing between tasks (vip or dnsrr). Defaults to vip if not provided.
  - **Ports** – List of exposed ports that this service is accessible on from the outside, in the form of: {"Protocol": <"tcp"|"udp">, "PublishedPort": <port>, "TargetPort": <port>}. Ports can only be provided if vip resolution mode is used.

Request Headers:

- **Content-type** – Set to "application/json".
- **X-Registry-Auth** – base64-encoded AuthConfig object, containing either login information, or a token. Refer to the create an image section for more details.

REMOVE A SERVICE

DELETE /services/(id or name)

Stop and remove the service id

Example request:

```
DELETE /v1.24/services/16253994b7c4 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

Status codes:

- **200** – no error
- **404** – no such service
- **406** - node is not part of a swarm
- **500** – server error

INSPECT ONE OR MORE SERVICES

GET /services/(id or name)

Return information on the service id.

Example request:

```
GET /v1.24/services/1cb4dnqcyx6m66g2t538x3rxha HTTP/1.1
```

Example response:

```
{
  "ID": "ak7w3gjqoa3kuz8xcpnyy0pvl",
  "Version": {
    "Index": 95
  },
"CreatedAt": "2016-06-07T21:10:20.269723157Z",
"UpdatedAt": "2016-06-07T21:10:20.276301259Z",
"Spec": {
  "Name": "redis",
  "TaskTemplate": {
    "ContainerSpec": {
      "Image": "redis"
    },
    "Resources": {
      "Limits": {},
      "Reservations": {}
    },
    "RestartPolicy": {
      "Condition": "any",
      "MaxAttempts": 0
    },
    "Placement": {}
  },
  "Mode": {
    "Replicated": {
      "Replicas": 1
    }
  },
  "UpdateConfig": {
    "Parallelism": 1,
    "FailureAction": "pause"
  },
  "EndpointSpec": {
    "Mode": "vip",
    "Ports": [
      {
        "Protocol": "tcp",
        "TargetPort": 6379,
        "PublishedPort": 30001
      }
    ]
  }
},
"Endpoint": {
  "Spec": {
    "Mode": "vip",
    "Ports": [
      {
```

```
                "Protocol": "tcp",
                "TargetPort": 6379,
                "PublishedPort": 30001
            }
        ]
    },
    "Ports": [
        {
            "Protocol": "tcp",
            "TargetPort": 6379,
            "PublishedPort": 30001
        }
    ],
    "VirtualIPs": [
        {
            "NetworkID": "4qvuz4ko70xaltuqbt8956gd1",
            "Addr": "10.255.0.4/16"
        }
    ]
  }
}
```

Status codes:

- **200** – no error
- **404** – no such service
- **406** - node is not part of a swarm
- **500** – server error

UPDATE A SERVICE

POST /services/(id)/update

Update a service. When using this endpoint to create a service using a private repository from the registry, the X-Registry-Auth header can be used to update the authentication information for that is stored for the service. The header contains a base64-encoded AuthConfig object. Refer to the create an image section for more details.

Example request:

```
POST /v1.24/services/1cb4dnqcyx6m66g2t538x3rxha/update?version=23 HTTP/1.1
Content-Type: application/json
Content-Length: 12345

{
  "Name": "top",
  "TaskTemplate": {
    "ContainerSpec": {
        "Image": "busybox",
        "Args": [
          "top"
        ]
      },
      "Resources": {
        "Limits": {},
        "Reservations": {}
      },
      "RestartPolicy": {
        "Condition": "any",
        "MaxAttempts": 0
      },
      "Placement": {}
    },
    "Mode": {
      "Replicated": {
        "Replicas": 1
      }
    },
    "UpdateConfig": {
      "Parallelism": 1
    },
    "EndpointSpec": {
      "Mode": "vip"
    }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: text/plain; charset=utf-8
```

JSON Parameters:

- **Name** – User-defined name for the service. Note that renaming services is not supported.
- **Labels** – A map of labels to associate with the service (e.g., {"key":"value", "key2":"value2"}).
- **TaskTemplate** – Specification of the tasks to start as part of the new service.
  - **ContainerSpec** - Container settings for containers started as part of this task.
    - **Image** – A string specifying the image name to use for the container.
    - **Command** – The command to be run in the image.
    - **Args** – Arguments to the command.
    - **Env** – A list of environment variables in the form of ["VAR=value"[,"VAR2=value2"]].
    - **Dir** – A string specifying the working directory for commands to run in.
    - **User** – A string value specifying the user inside the container.

        - **Labels** – A map of labels to associate with the service (e.g.,{"key":"value", "key2":"value2"}).
        - **Mounts** – Specification for mounts to be added to containers created as part of the new service.
            - **Target** – Container path.
            - **Source** – Mount source (e.g. a volume name, a host path).
            - **Type** – The mount type (bind, or volume).
            - **ReadOnly** – A boolean indicating whether the mount should be read-only.
            - **BindOptions** - Optional configuration for the bind type
                - **Propagation** – A propagation mode with the value [r]private, [r]shared, or [r]slave.
            - **VolumeOptions** – Optional configuration for the volume type.
                - **NoCopy** – A boolean indicating if volume should be populated with the data from the target. (Default false)
                - **Labels** – User-defined name and labels for the volume.
                - **DriverConfig** – Map of driver-specific options.
                    - **Name** - Name of the driver to use to create the volume
                    - **Options** - key/value map of driver specific options
        - **StopGracePeriod** – Amount of time to wait for the container to terminate before forcefully killing it.
    - **Resources** – Resource requirements which apply to each individual container created as part of the service.
        - **Limits** – Define resources limits.
            - **CPU** – CPU limit
            - **Memory** – Memory limit
        - **Reservation** – Define resources reservation.
            - **CPU** – CPU reservation
            - **Memory** – Memory reservation
    - **RestartPolicy** – Specification for the restart policy which applies to containers created as part of this service.
        - **Condition** – Condition for restart (none, on-failure, or any).
        - **Delay** – Delay between restart attempts.
        - **MaxAttempts** – Maximum attempts to restart a given container before giving up (default value is 0, which is ignored).
        - **Window** – Windows is the time window used to evaluate the restart policy (default value is 0, which is unbounded).
    - **Placement** – Restrictions on where a service can run.
        - **Constraints** – An array of constraints, e.g. ["node.role == manager"].
- **Mode** – Scheduling mode for the service (replicated or global, defaults to replicated).
- **UpdateConfig** – Specification for the update strategy of the service.
    - **Parallelism** – Maximum number of tasks to be updated in one iteration (0 means unlimited parallelism).
    - **Delay** – Amount of time between updates.
- **Networks** – Array of network names or IDs to attach the service to.

- **EndpointSpec** – Properties that can be configured to access and load balance a service.
    - **Mode** – The mode of resolution to use for internal load balancing between tasks (vip or dnsrr). Defaults to vip if not provided.
    - **Ports** – List of exposed ports that this service is accessible on from the outside, in the form of: {"Protocol": <"tcp"|"udp">, "PublishedPort": <port>, "TargetPort": <port>}. Ports can only be provided if vip resolution mode is used.

Query parameters:

- **version** – The version number of the service object being updated. This is required to avoid conflicting writes.

Request Headers:

- **Content-type** – Set to "application/json".
- **X-Registry-Auth** – base64-encoded AuthConfig object, containing either login information, or a token. Refer to the create an image section for more details.

Status codes:

- **200** – no error
- **404** – no such service
- **406** - node is not part of a swarm
- **500** – server error

3.10 Tasks

Note: Task operations require the engine to be part of a swarm.

LIST TASKS

GET /tasks

List tasks

Example request:

```
GET /v1.24/tasks HTTP/1.1
```

Example response:

```
[
  {
    "ID": "0kzzo1i0y4jz6027t0k7aezc7",
    "Version": {
      "Index": 71
    },
    "CreatedAt": "2016-06-07T21:07:31.171892745Z",
    "UpdatedAt": "2016-06-07T21:07:31.376370513Z",
    "Spec": {
      "ContainerSpec": {
        "Image": "redis"
      },
      "Resources": {
        "Limits": {},
        "Reservations": {}
      },
      "RestartPolicy": {
        "Condition": "any",
        "MaxAttempts": 0
      },
      "Placement": {}
    },
    "ServiceID": "9mnpnzenvg8p8tdbtq4wvbkcz",
    "Slot": 1,
    "NodeID": "60gvrl6tm78dmak4yl7srz94v",
    "Status": {
      "Timestamp": "2016-06-07T21:07:31.290032978Z",
      "State": "running",
      "Message": "started",
      "ContainerStatus": {
        "ContainerID": "e5d62702a1b48d01c3e02ca1e0212a250801fa8d67caca0b6f35919ebc12f035",
        "PID": 677
      }
    },
    "DesiredState": "running",
    "NetworksAttachments": [
      {
        "Network": {
          "ID": "4qvuz4ko70xaltuqbt8956gd1",
          "Version": {
            "Index": 18
          },
          "CreatedAt": "2016-06-07T20:31:11.912919752Z",
          "UpdatedAt": "2016-06-07T21:07:29.955277358Z",
          "Spec": {
            "Name": "ingress",
            "Labels": {
              "com.docker.swarm.internal": "true"
            },
            "DriverConfiguration": {},
            "IPAMOptions": {
              "Driver": {},
              "Configs": [
                {
                  "Subnet": "10.255.0.0/16",
                  "Gateway": "10.255.0.1"
                }
```

```
                    ]
                  }
                },
                "DriverState": {
                  "Name": "overlay",
                  "Options": {
                    "com.docker.network.driver.overlay.vxlanid_list": "256"
                  }
                },
                "IPAMOptions": {
                  "Driver": {
                    "Name": "default"
                  },
                  "Configs": [
                    {
                      "Subnet": "10.255.0.0/16",
                      "Gateway": "10.255.0.1"
                    }
                  ]
                }
              },
              "Addresses": [
                "10.255.0.10/16"
              ]
            }
          ],
        },
        {
          "ID": "1yljwbmlr8er2waf8orvqpwms",
          "Version": {
            "Index": 30
          },
          "CreatedAt": "2016-06-07T21:07:30.019104782Z",
          "UpdatedAt": "2016-06-07T21:07:30.231958098Z",
          "Name": "hopeful_cori",
          "Spec": {
            "ContainerSpec": {
              "Image": "redis"
            },
            "Resources": {
              "Limits": {},
              "Reservations": {}
            },
    "RestartPolicy": {
      "Condition": "any",
      "MaxAttempts": 0
    },
    "Placement": {}
  },
  "ServiceID": "9mnpnzenvg8p8tdbtq4wvbkcz",
  "Slot": 1,
  "NodeID": "60gvrl6tm78dmak4yl7srz94v",
  "Status": {
    "Timestamp": "2016-06-07T21:07:30.202183143Z",
    "State": "shutdown",
    "Message": "shutdown",
    "ContainerStatus": {
      "ContainerID": "1cf8d63d18e79668b0004a4be4c6ee58cddfad2dae29506d8781581d0688a213"
    }
  },
  "DesiredState": "shutdown",
  "NetworksAttachments": [
    {
      "Network": {
        "ID": "4qvuz4ko70xaltuqbt8956gd1",
        "Version": {
          "Index": 18
        },
        "CreatedAt": "2016-06-07T20:31:11.912919752Z",
        "UpdatedAt": "2016-06-07T21:07:29.955277358Z",
        "Spec": {
          "Name": "ingress",
          "Labels": {
            "com.docker.swarm.internal": "true"
          },
          "DriverConfiguration": {},
          "IPAMOptions": {
            "Driver": {},
            "Configs": [
              {
                "Subnet": "10.255.0.0/16",
                "Gateway": "10.255.0.1"
              }
            ]
          }
        },
            "DriverState": {
              "Name": "overlay",
              "Options": {
                "com.docker.network.driver.overlay.vxlanid_list": "256"
              }
            },
            "IPAMOptions": {
              "Driver": {
                "Name": "default"
              },
              "Configs": [
                {
                  "Subnet": "10.255.0.0/16",
                  "Gateway": "10.255.0.1"
                }
              ]
            }
          },
          "Addresses": [
            "10.255.0.5/16"
          ]
        }
      ]
    }
]
```

Query parameters:

- **filters** – a JSON encoded value of the filters (a map[string][]string) to process on the services list. Available filters:
  - id=<task id>
  - name=<task name>
  - service=<service name>
  - node=<node id or name>
  - label=key or label="key=value"
  - desired-state=(running | shutdown | accepted)

Status codes:

- **200** – no error
- **406** - node is not part of a swarm
- **500** – server error

INSPECT A TASK

GET /tasks/(id)

Get details on the task id

Example request:

```
GET /v1.24/tasks/0kzzo1i0y4jz6027t0k7aezc7 HTTP/1.1
```

Example response:

```
{
  "ID": "0kzzo1i0y4jz6027t0k7aezc7",
  "Version": {
    "Index": 71
  },
  "CreatedAt": "2016-06-07T21:07:31.171892745Z",
  "UpdatedAt": "2016-06-07T21:07:31.376370513Z",
  "Spec": {
    "ContainerSpec": {
      "Image": "redis"
    },
    "Resources": {
      "Limits": {},
      "Reservations": {}
    },
    "RestartPolicy": {
      "Condition": "any",
      "MaxAttempts": 0
    },
    "Placement": {}
  },
  "ServiceID": "9mnpnzenvg8p8tdbtq4wvbkcz",
  "Slot": 1,
  "NodeID": "60gvrl6tm78dmak4yl7srz94v",
  "Status": {
    "Timestamp": "2016-06-07T21:07:31.290032978Z",
    "State": "running",
    "Message": "started",
    "ContainerStatus": {
      "ContainerID": "e5d62702a1b48d01c3e02ca1e0212a250801fa8d67caca0b6f35919ebc12f035",
      "PID": 677
    }
  },
  "DesiredState": "running",
  "NetworksAttachments": [
    {
      "Network": {
        "ID": "4qvuz4ko70xaltuqbt8956gd1",
        "Version": {
          "Index": 18
        },
        "CreatedAt": "2016-06-07T20:31:11.912919752Z",
        "UpdatedAt": "2016-06-07T21:07:29.955277358Z",
        "Spec": {
          "Name": "ingress",
          "Labels": {
            "com.docker.swarm.internal": "true"
          },
          "DriverConfiguration": {},
          "IPAMOptions": {
            "Driver": {},
            "Configs": [
              {
                "Subnet": "10.255.0.0/16",
                "Gateway": "10.255.0.1"
              }
            ]
          }
        },
        "DriverState": {
          "Name": "overlay",
          "Options": {
            "com.docker.network.driver.overlay.vxlanid_list": "256"
          }
        },
        "IPAMOptions": {
          "Driver": {
            "Name": "default"
          },
          "Configs": [
            {
              "Subnet": "10.255.0.0/16",
              "Gateway": "10.255.0.1"
            }
          ]
        }
      },
      "Addresses": [
        "10.255.0.10/16"
      ]
    }
  ]
}
```

Status codes:

- **200** – no error
- **404** – unknown task
- **406** - node is not part of a swarm
- **500** – server error

4. Going further

4.1 Inside docker run

As an example, the docker run command line makes the following API calls:

- Create the container
  - If the status code is 404, it means the image doesn't exist:
    - Try to pull it.
    - Then, retry to create the container.

      Start the container.
    -
  - If you are not in detached mode:
    -

    Attach to the container, using logs=1 (to have stdout and stderr from the container's start) and stream=1
    -
  - If in detached mode or only stdin is attached, display the container's id.

4.2 Hijacking

In this version of the API, /attach, uses hijacking to transport stdin, stdout, and stderr on the same socket.

To hint potential proxies about connection hijacking, Docker client sends connection upgrade headers similarly to websocket.

```
Upgrade: tcp
Connection: Upgrade
```

When Docker daemon detects the Upgrade header, it switches its status code from **200 OK** to **101 UPGRADED** and resends the same headers.

4.3 CORS Requests

To set cross origin requests to the Engine API please give values to --api-cors-header when running Docker in daemon mode. Set * (asterisk) allows all, default or blank means CORS disabled

```
$ dockerd -H="192.168.1.9:2375" --api-cors-header="http://foo.bar"
```