Starting from version `0.2.0`, Harvester can be installed automatically. This document provides an example to do an automatic installation with PXE boot.

We recommend using iPXE to perform the network boot. It has more features than the traditional PXE Boot program and is likely available in modern NIC cards. If the iPXE firmware is not available for your NIC card, the iPXE firmware images can be loaded from the TFTP server first.

To see sample iPXE scripts, please visit Harvester iPXE Examples.

Prerequisite

Nodes need to have at least **8 GiB** of RAM because the installer loads the full ISO file into tmpfs.

:::info The installer automatically checks if the hardware meets the minimum requirements for production use. If any of the checks fail, installation will be stopped. To override this behavior, set either the configuration file option `install.skipchecks=true` or the kernel parameter `harvester.install.skipchecks=true`. :::

Preparing HTTP Servers

An HTTP server is required to serve boot files. Let's assume the NGINX HTTP server's IP is `10.100.0.10`, and it serves the `/usr/share/nginx/html/` directory with the path `http://10.100.0.10/`.

Preparing Boot Files

- Download the required files from the Harvester releases page.
  - The ISO: `harvester-<version>-amd64.iso`
  - The kernel: `harvester-<version>-vmlinuz-amd64`
  - The initrd: `harvester-<version>-initrd-amd64`
  - The rootfs squashfs image: `harvester-<version>-rootfs-amd64.squashfs`
- Serve the files.

  Copy or move the downloaded files to an appropriate location so they can be downloaded via the HTTP server. For example:

  ```
  sudo mkdir -p /usr/share/nginx/html/harvester/
  sudo cp /path/to/harvester-<version>-amd64.iso /usr/share/nginx/html/harvester/
  sudo cp /path/to/harvester-<version>-vmlinuz-amd64
  /usr/share/nginx/html/harvester/
  sudo cp /path/to/harvester-<version>-initrd-amd64
  /usr/share/nginx/html/harvester/
  sudo cp /path/to/harvester-<version>-rootfs-amd64.squashfs
  /usr/share/nginx/html/harvester/
  ```

Preparing iPXE Boot Scripts

When performing an automatic installation, there are two modes:

- `CREATE` : we are installing a node to construct an initial Harvester cluster.
- `JOIN` : we are installing a node to join an existing Harvester cluster.

Harvester v1.3.0 introduces roles that you can assign to nodes to support different scenarios. For more information, see Harvester Configuration.

CREATE Mode

:::caution

Security Risks: The configuration file below contains credentials which should be kept secret. Please do not make the configuration file publicly accessible.

:::

Create a Harvester configuration file called `config-create.yaml` for `CREATE` mode. Modify the values as needed:

```
# cat /usr/share/nginx/html/harvester/config-create.yaml
scheme_version: 1
token: token # Replace with a desired token
os:
  hostname: node1 # Set a hostname. This can be omitted if DHCP server offers
hostnames
  ssh_authorized_keys:
  - ssh-rsa ... # Replace with your public key
  password: p@ssword     # Replace with your password
  ntp_servers:
  - 0.suse.pool.ntp.org
  - 1.suse.pool.ntp.org
install:
  mode: create
  management_interface: # available as of v1.1.0
    interfaces:
      - name: ens5
    default_route: true
    method: dhcp
    bond_options:
      mode: balance-tlb
      miimon: 100
  device: /dev/sda # The target disk to install
#  data_disk: /dev/sdb # It is recommended to use a separate disk to store VM data
  iso_url: http://10.100.0.10/harvester/harvester-<version>-amd64.iso
#  tty: ttyS1,115200n8   # For machines without a VGA console

  vip: 10.100.0.99        # The VIP to access the Harvester GUI. Make sure the IP is
free to use
  vip_mode: static        # Or dhcp, check configuration file for more information
#  vip_hw_addr: 52:54:00:ec:0e:0b   # Leave empty when vip_mode is static
```

For machines that needs to be installed using `CREATE` mode, the following is an iPXE script that boots the kernel with the above config:

```
#!ipxe
kernel harvester-<version>-vmlinuz ip=dhcp net.ifnames=1 rd.cos.disable rd.noverifyssl
```

```
console=tty1 root=live:http://10.100.0.10/harvester/rootfs.squashfs
harvester.install.automatic=true
harvester.install.config_url=http://10.100.0.10/harvester/config-create.yaml
initrd harvester-<version>-initrd
boot
```

This assumes the iPXE script is stored in `/usr/share/nginx/html/harvester/ipxe-create`.

:::note

If you have multiple network interfaces, you can leverage dracut's `ip=` parameter to specify the booting interface and any other network configurations that dracut supports (e.g., `ip=eth1:dhcp`). See [man `dracut.cmdline`] for more information.

Use `ip=` parameter to designate the booting interface only, as we only support **one single `ip=` parameter**.

:::

JOIN Mode

:::caution

Security Risks: The configuration file below contains credentials which should be kept secret. Please do not make the configuration file publicly accessible.

:::

Create a Harvester configuration file called `config-join.yaml` for `JOIN` mode. Modify the values as needed:

```
# cat /usr/share/nginx/html/harvester/config-join.yaml
scheme_version: 1
server_url: https://10.100.0.99:443  # Should be the VIP set up in "CREATE" config
token: token
os:
  hostname: node2
  ssh_authorized_keys:
    - ssh-rsa ... # Replace with your public key
  password: p@ssword     # Replace with your password
  dns_nameservers:
  - 1.1.1.1
  - 8.8.8.8
install:
  mode: join
  management_interface: # available as of v1.1.0
    interfaces:
      - name: ens5
    default_route: true
    method: dhcp
    bond_options:
      mode: balance-tlb
      miimon: 100
  device: /dev/sda # The target disk to install
```

```
#  data_disk: /dev/sdb # It is recommended to use a separate disk to store VM data
  iso_url: http://10.100.0.10/harvester/harvester-<version>-amd64.iso
#  tty: ttyS1,115200n8   # For machines without a VGA console
```

Note that the `mode` is `join` and the `server_url` needs to be provided.

For machines that needs to be installed in `JOIN` mode, the following is an iPXE script that boots the kernel with the above config:

```
#!ipxe
kernel harvester-<version>-vmlinuz ip=dhcp net.ifnames=1 rd.cos.disable rd.noverifyssl
console=tty1 root=live:http://10.100.0.10/harvester/rootfs.squashfs
harvester.install.automatic=true
harvester.install.config_url=http://10.100.0.10/harvester/config-join.yaml
initrd harvester-<version>-initrd
boot
```

This assumes the iPXE script is stored in `/usr/share/nginx/html/harvester/ipxe-join`.

DHCP Server Configuration

:::note

In the PXE installation scenario, you are required to add the *routers* option (`option routers`) when configuring the DHCP server. This option is used to add the default route on the Harvester host. Without the default route, the node will fail to start.

In the ISO installation scenario, when the management network interface is in DHCP mode, you are also required to add the *routers* option (`option routers`) when configuring the DHCP server.

For example:

```
    Harvester Host:~ # ip route
    default via 192.168.122.1 dev mgmt-br proto dhcp
```

For more information, see ISC DHCPv4 Option Configuration.

:::

The following is an example of how to configure the ISC DHCP server to offer iPXE scripts:

```
option architecture-type code 93 = unsigned integer 16;

subnet 10.100.0.0 netmask 255.255.255.0 {
    option routers 10.100.0.10;
        option domain-name-servers 192.168.2.1;
    range 10.100.0.100 10.100.0.253;
}

group {
  # create group
  if exists user-class and option user-class = "iPXE" {
    # iPXE Boot
    if option architecture-type = 00:07 {
      filename "http://10.100.0.10/harvester/ipxe-create-efi";
    } else {
      filename "http://10.100.0.10/harvester/ipxe-create";
    }
  } else {
    # PXE Boot
    if option architecture-type = 00:07 {
      # UEFI
      filename "ipxe.efi";
    } else {
      # Non-UEFI
      filename "undionly.kpxe";
    }
  }

  host node1 { hardware ethernet 52:54:00:6b:13:e2; }
}

group {
  # join group
  if exists user-class and option user-class = "iPXE" {
    # iPXE Boot
    if option architecture-type = 00:07 {
      filename "http://10.100.0.10/harvester/ipxe-join-efi";
    } else {
      filename "http://10.100.0.10/harvester/ipxe-join";
    }
  } else {
    # PXE Boot
    if option architecture-type = 00:07 {
      # UEFI
      filename "ipxe.efi";
    } else {
      # Non-UEFI
      filename "undionly.kpxe";
    }
  }

  host node2 { hardware ethernet 52:54:00:69:d5:92; }
}
```

The config file declares a subnet and two groups. The first group is for hosts to boot using `CREATE` mode and the other one is for `JOIN` mode. By default, the iPXE path is chosen, but if it sees a PXE client it offers the iPXE image according to the client architecture. Please prepare those images and a TFTP server first.

The Internet Systems Consortium (ISC) announced the final end-of-life (EOL) for ISC DHCP in 2022. ISC DHCP users are encouraged to migrate to the newer, feature-rich Kea DHCP, which the ISC designed for more modern network environments. If you are already using the Kea DHCPv4 server, check the following configuration example. For more information, see Kea DHCPv4 Configuration.

```
"client-classes": [
  {
    "name": "iPXE UEFI/CREATE",
    "test": "option[user-class].exists and substring(option[user-class].hex,0,4) ==
'iPXE' and option[client-system].hex == 0x0007",
    "boot-file-name": "http://10.100.0.10/harvester/ipxe-create-efi",
    "only-if-required": true
  },
  {
    "name": "iPXE non-UEFI/CREATE",
    "test": "option[user-class].exists and substring(option[user-class].hex,0,4) ==
'iPXE' and not option[client-system].hex == 0x0007",
    "boot-file-name": "http://10.100.0.10/harvester/ipxe-create",
    "only-if-required": true
  },
  {
    "name": "iPXE UEFI/JOIN",
    "test": "option[user-class].exists and substring(option[user-class].hex,0,4) ==
'iPXE' and option[client-system].hex == 0x0007",
    "boot-file-name": "http://10.100.0.10/harvester/ipxe-join-efi",
    "only-if-required": true
  },
  {
    "name": "iPXE non-UEFI/JOIN",
    "test": "option[user-class].exists and substring(option[user-class].hex,0,4) ==
'iPXE' and not option[client-system].hex == 0x0007",
    "boot-file-name": "http://10.100.0.10/harvester/ipxe-join",
    "only-if-required": true
  },
  {
    "name": "PXE UEFI",
    "test": "option[user-class].exists and not substring(option[user-class].hex,0,4)
== 'iPXE' and option[client-system].hex == 0x0007",
    "next-server": "10.100.0.20",
    "boot-file-name": "ipxe.efi"
  },
  {
    "name": "PXE non-UEFI",
    "test": "option[user-class].exists and not substring(option[user-class].hex,0,4)
== 'iPXE' and option[client-system].hex == 0x0007",
    "next-server": "10.100.0.20",
    "boot-file-name": "undionly.kpxe"
  }
]

"subnet4": [
  {
    "subnet": "10.100.0.0/24",
    "pools": [
      {
        "pool": "10.100.0.100 - 10.100.0.199",
        "require-client-classes" : [ "iPXE UEFI/CREATE", "iPXE non-UEFI/CREATE" ]
      }.
      {
        "pool": "10.100.0.200 - 10.100.0.253",
        "require-client-classes" : [ "iPXE UEFI/JOIN", "iPXE non-UEFI/JOIN" ]
      }
    ],
    "option-data": [
      {
        "name": "routers",
        "data": "10.100.0.10"
      }
    ],
    "reservations": [
      // assign ip address to the host for booting in CREATE mode
      {
        "hw-address": "52:54:00:6b:13:e2",
        "ip-address": "10.100.0.101"
      },
      // assign ip address to the host for booting in JOIN mode
      {
        "hw-address": "52:54:00:69:d5:92",
        "ip-address": "10.100.0.201"
      }
    ]
  }
]
```

Harvester Configuration

For more information about Harvester configuration, please refer to the Harvester configuration page.

By default, the first node will be the management node of the cluster. When there are 3 nodes, the other 2 nodes added first are automatically promoted to management nodes to form an HA cluster.

If you want to promote management nodes from different zones, you can add the node label `topology.kubernetes.io/zone` in the os.labels config. In this case, at least three different zones are required.

Users can also provide configuration via kernel parameters. For example, to specify the `CREATE` install mode, users can pass the `harvester.install.mode=create` kernel parameter when booting. Values passed through kernel parameters have higher priority than values specified in the config file.

UEFI HTTP Boot support

UEFI firmware supports loading a boot image from an HTTP server. This section demonstrates how to use UEFI HTTP boot to load the iPXE program and perform an automatic installation.

Serve the iPXE Program

Download the iPXE UEFI program from http://boot.ipxe.org/ipxe.efi and make sure `ipxe.efi` can be downloaded from the HTTP server. For example:

```
cd /usr/share/nginx/html/harvester/
wget http://boot.ipxe.org/ipxe.efi
```

The file now can be downloaded from http://10.100.0.10/harvester/ipxe.efi locally.

DHCP Server Configuration

If the user plans to use the UEFI HTTP boot feature by getting a dynamic IP first, the DHCP server needs to provide the iPXE program URL when it sees such a request. The following is an updated ISC DHCP server group example:

```
group {
  # create group
  if exists user-class and option user-class = "iPXE" {
    # iPXE Boot
    if option architecture-type = 00:07 {
      filename "http://10.100.0.10/harvester/ipxe-create-efi";
    } else {
      filename "http://10.100.0.10/harvester/ipxe-create";
    }
  } elsif substring (option vendor-class-identifier, 0, 10) = "HTTPClient" {
    # UEFI HTTP Boot
    option vendor-class-identifier "HTTPClient";
    filename "http://10.100.0.10/harvester/ipxe.efi";
  } else {
    # PXE Boot
    if option architecture-type = 00:07 {
      # UEFI
      filename "ipxe.efi";
    } else {
      # Non-UEFI
      filename "undionly.kpxe";
    }
  }

  host node1 { hardware ethernet 52:54:00:6b:13:e2; }
}
```

The `elsif substring` statement is new, and it offers `http://10.100.0.10/harvester/ipxe.efi` when it sees a UEFI HTTP boot DHCP request. After the client fetches the iPXE program and runs it, the iPXE program will send a DHCP request again and load the iPXE script from the URL `http://10.100.0.10/harvester/ipxe-create-efi`.

If you want to enable UEFI HTTP boot on the Kea DHCPv4 server, you must add a new `client-class` at the end of the `client-classes`.

Example:

```
{
  "name": "HTTP",
  "test": "substring(option[vendor-class-identifier].hex,0,10) == 'HTTPClient'",
  "option-data": [
    {
      "name": "vendor-class-identifier",
      "data": "HTTPClient"
    }
  ],
  "boot-file-name": "http://10.100.0.10/harvester/ipxe.efi"
}
```

The iPXE Script for UEFI Boot

It's mandatory to specify the initrd image for UEFI boot in the kernel parameters. The following is an updated version of iPXE script for `CREATE` mode.

```
#!ipxe
kernel harvester-<version>-vmlinuz initrd=harvester-<version>-initrd ip=dhcp
net.ifnames=1 rd.cos.disable rd.noverifyssl console=tty1
root=live:http://10.100.0.10/harvester/rootfs.squashfs
harvester.install.automatic=true
harvester.install.config_url=http://10.100.0.10/harvester/config-create.yaml
initrd harvester-<version>-initrd
boot
```

The parameter `initrd=harvester-<version>-initrd` is required.

Useful Kernel Parameters

Besides the Harvester configuration, you can also specify other kernel parameters that are useful in different scenarios. See also dracut.cmdline(7).

`ip=dhcp`

If you have multiple network interfaces, you could add the `ip=dhcp` parameter to get IP from the DHCP server from all interfaces.

`rd.net.dhcp.retry=<cnt>`

Failing to get IP from the DHCP server would cause iPXE booting to fail. You can add parameter `rd.net.dhcp.retry=<cnt>` to retry DHCP request for `<cnt>` times.

`harvester.install.skipchecks=true`

Installation is stopped if the hardware checks fail (because the minimum requirements for production use are not met). To override this behavior, set the kernel parameter `harvester.install.skipchecks=true`. When set to `true`, warning messages are still saved to `/var/log/console.log`, but the installation proceeds even if hardware requirements for production use are not met.

`harvester.install.with_net_images=true`

The installer does not preload images during installation and instead pulls all required images from the internet after installation is completed. Usage of this parameter is not recommended in most cases. For more information, see Net Install ISO.