



Grok 3 ▾



Write a script for me using ansible for enabling PXE boot using remote BMC LAN connection. It should be set to use specific interface and boot from - if it is possible - from specific address - 10.123.232.180.

Below is an Ansible playbook script that enables PXE boot using a remote BMC (Baseboard Management Controller) LAN connection, configures it to use a specific network interface, and, where possible, sets the boot source to a specific PXE server address (10.123.232.180). This script assumes the use of IPMI (Intelligent Platform Management Interface) for BMC communication, as it's a common standard for managing server hardware remotely. Note that the ability to specify a particular PXE server address (like 10.123.232.180) depends on the BMC firmware and its support for such configurations, which is not universally guaranteed.

The playbook uses the `community.general.ipmi_boot` module to interact with the BMC and configure boot settings. It also includes tasks to ensure the specified network interface is used for PXE booting, assuming the BMC allows interface selection. If the BMC does not support setting a specific PXE server address, the script will configure PXE as the boot method and rely on the network's DHCP/TFTP setup to direct to the desired server.

Ansible Playbook: `enable_pxe_boot.yml`

```
---
- name: Configure PXE boot via BMC LAN connection
  hosts: bmc_servers
  gather_facts: no
  vars:
    bmc_user: "admin"
    bmc_password: "password"
    bmc_interface: "eth0" # Specify the network interface for PXE
    pxe_server_ip: "10.123.232.180" # Desired PXE server IP
    boot_timeout: 300 # Timeout for boot configuration in seconds

  tasks:
    - name: Ensure IPMI tool is installed on the control node
      ansible.builtin.package:
        name: ipmitool
        state: present
      delegate_to: localhost
      become: yes

    - name: Configure BMC to enable PXE boot
      community.general.ipmi_boot:
        name: "{{ inventory_hostname }}"
```



Grok 3 ▾



```
    user: "{{ bmc_user }}"
    password: "{{ bmc_password }}"
    bootdev: pxe
    lan_interface: "{{ bmc_interface }}"
    state: present
  register: pxe_boot_result
  retries: 3
  delay: 10
  until: pxe_boot_result is success

- name: Verify PXE boot configuration
  ansible.builtin.command:
    cmd: ipmitool -I lanplus -H {{ inventory_hostname }} -U {{ bmc_user }} -P {{ bmc_password }}
  register: bootparam_output
  changed_when: false
  failed_when: "'Boot Device Selector : PXE' not in bootparam_output.stdout"

- name: Attempt to set PXE server address (if supported)
  ansible.builtin.command:
    cmd: ipmitool -I lanplus -H {{ inventory_hostname }} -U {{ bmc_user }} -P {{ bmc_password }}
  register: pxe_server_result
  ignore_errors: yes # Not all BMCs support setting specific PXE server IP
  when: pxe_boot_result is success

- name: Reboot server to apply PXE boot settings
  community.general.ipmi_power:
    name: "{{ inventory_hostname }}"
    user: "{{ bmc_user }}"
    password: "{{ bmc_password }}"
    state: boot
  when: pxe_boot_result is success

- name: Wait for server to become reachable after reboot
  ansible.builtin.wait_for:
    host: "{{ inventory_hostname }}"
    port: 22
    state: started
    timeout: "{{ boot_timeout }}"
  delegate_to: localhost
  when: pxe_boot_result is success
  ignore_errors: yes
```

Explanation of the Playbook

• Playbook Structure:

- **Hosts:** Targets a group called `bmc_servers`, which should be defined in your Ansible inventory with the BMC IP addresses.

- ×

Gather Facts: Disabled (`gather_facts: no`) since we're interacting with BMCs, not the OS.

Grok 3

↑

🔖

🔄

✎

yaml

📄

 - **Variables:**
 - `bmc_user` and `bmc_password`: Credentials for BMC access (replace with secure values, preferably using Ansible Vault).
 - `bmc_interface`: The network interface (e.g., `eth0`) to use for PXE booting.
 - `pxe_server_ip`: The desired PXE server IP (10.123.232.180).
 - `boot_timeout`: Time to wait for the server to reboot and become reachable.
 - **Tasks:**
 - **Install IPMI Tool:** Ensures `ipmitool` is installed on the Ansible control node for BMC communication.
 - **Configure PXE Boot:** Uses the `community.general.ipmi_boot` module to set the boot device to PXE and specify the network interface (`lan_interface`). Retries up to 3 times with a 10-second delay to handle transient BMC issues.
 - **Verify PXE Boot Configuration:** Runs `ipmitool chassis bootparam get 5` to confirm PXE is set as the boot device.
 - **Set PXE Server Address:** Attempts to configure the PXE server IP using a raw IPMI command. This is vendor-specific and may not be supported by all BMCs, so errors are ignored (`ignore_errors: yes`).
 - **Reboot Server:** Uses `community.general.ipmi_power` to reboot the server and apply the PXE boot settings.
 - **Wait for Server:** Waits for the server to become reachable on port 22 (SSH) after rebooting, with a timeout to avoid hanging.
- ## Prerequisites
- **Ansible Setup:**
- Install Ansible and the `community.general` collection:
- ```
ansible-galaxy collection install community.general
bash
```
- Ensure `ipmitool` is installed on the control node (handled by the playbook).
- **Inventory File:**
- Create an Ansible inventory file (e.g., `inventory.yml`) with the BMC IP addresses:
- ```
bmc_servers:
  hosts:
    bmc1:
      ansible_host: 192.168.1.100
```
- ×

bmc2:

ansible_host: 192.168.1.101

Grok 3

↑

🔖

🔄

✎

yaml

📄

Replace `192.168.1.100` and `192.168.1.101` with your BMC IP addresses.

 - **Network Configuration:**
 - Ensure the BMC is accessible over the network via IPMI.
 - The PXE server at `10.123.232.180` must be running a DHCP/TFTP service configured to serve boot images.
 - The specified network interface (`eth0`) must be connected and configured for PXE.
 - **BMC Compatibility:**
 - The BMC must support IPMI and PXE boot configuration.
 - Setting a specific PXE server IP is hardware-dependent. Check your BMC documentation (e.g., Dell iDRAC, HPE iLO, Supermicro BMC) for support. If unsupported, the server will use the DHCP-provided PXE server.
- ## How to Run the Playbook
- Save the playbook as `enable_pxe_boot.yml`.
 - Update the inventory file with your BMC IP addresses.
 - Replace `admin` and `password` with your BMC credentials (use Ansible Vault for security).
 - Run the playbook:
- ```
ansible-playbook -i inventory.yml enable_pxe_boot.yml
bash
```
- ## Notes and Limitations
- **PXE Server IP:** Many BMCs do not allow specifying a PXE server IP directly. In such cases, the server relies on the DHCP server to provide the PXE server details. Ensure your DHCP server is configured to point to `10.123.232.180`.
  - **Interface Selection:** The `lan_interface` parameter assumes the BMC supports selecting a specific NIC for PXE. If not, the BMC will use its default interface.
  - **Security:** Store BMC credentials securely using Ansible Vault instead of plaintext.
  - **Vendor Variability:** IPMI commands and PXE configuration options vary by vendor (e.g., Dell, HPE, Supermicro). Test the playbook on a single server first.
  - **Error Handling:** The playbook ignores errors for the PXE server IP task, as it's not universally supported. Check the `pxe_server_result` output for details.

Troubleshooting

- Grok 3
- **BMC Unreachable:** Verify the BMC IP, credentials, and network connectivity.
  - **PXE Boot Fails:** Ensure the PXE server ( 10.123.232.180 ) is running and reachable, and check DHCP/TFTP logs.
  - **Interface Not Used:** Confirm the BMC supports interface selection and that eth0 is correct.
  - **IPMI Command Fails:** Check BMC firmware compatibility and update if necessary.

If you need further customization (e.g., specific vendor commands, additional error handling, or integration with a specific PXE server setup), let me know your hardware details or requirements!

Reiterate above and make it better.

Below is an improved and streamlined Ansible playbook for enabling PXE boot via a remote BMC (Baseboard Management Controller) LAN connection. This version enhances the previous script by incorporating better error handling, improved modularity, secure credential management, and more robust validation. It configures PXE boot on a specific network interface (e.g., eth0 ) and attempts to set the PXE server to a specific IP address ( 10.123.232.180 ), where supported by the BMC firmware. The playbook is designed to be reusable, maintainable, and adaptable to various BMC implementations.

Improved Ansible Playbook: enable\_pxe\_boot.yml

```

- name: Configure PXE boot via BMC LAN connection
 hosts: bmc_servers
 gather_facts: no
 vars:
 bmc_interface: "eth0" # Network interface for PXE boot
 pxe_server_ip: "10.123.232.180" # Desired PXE server IP
 boot_timeout: 300 # Timeout for server reboot in seconds
 ipmi_retries: 3 # Number of retries for IPMI commands
 ipmi_delay: 10 # Delay between retries in seconds

 vars_files:
 - vars/bmc_credentials.yml # Securely store BMC credentials

 pre_tasks:
 - name: Validate required variables
 ansible.builtin.assert:
 that:
 - bmc_user is defined
 - bmc_password is defined
 - bmc_interface is defined
```

```

- pxe_server_ip is defined
fail_msg: "Required variables (bmc_user, bmc_password, bmc_interface, pxe_server_ip) are not defined"

tasks:
 - name: Ensure IPMI tool is installed on control node
 ansible.builtin.package:
 name: ipmitool
 state: present
 delegate_to: localhost
 become: yes
 run_once: true

 - name: Check BMC connectivity
 ansible.builtin.command:
 cmd: ipmitool -I lanplus -H {{ inventory_hostname }} -U {{ bmc_user }} -P {{ bmc_password }} ipmi bmc status
 register: bmc_status
 changed_when: false
 failed_when: bmc_status.rc != 0
 retries: "{{ ipmi_retries }}"
 delay: "{{ ipmi_delay }}"
 until: bmc_status.rc == 0

 - name: Configure BMC for PXE boot
 community.general.ipmi_boot:
 name: "{{ inventory_hostname }}"
 user: "{{ bmc_user }}"
 password: "{{ bmc_password }}"
 bootdev: pxe
 lan_interface: "{{ bmc_interface }}"
 state: present
 register: pxe_boot_result
 retries: "{{ ipmi_retries }}"
 delay: "{{ ipmi_delay }}"
 until: pxe_boot_result is success
 notify: Reboot server

 - name: Verify PXE boot configuration
 ansible.builtin.command:
 cmd: ipmitool -I lanplus -H {{ inventory_hostname }} -U {{ bmc_user }} -P {{ bmc_password }} ipmi boot param get
 register: bootparam_output
 changed_when: false
 failed_when: "'PXE' not in bootparam_output.stdout"
 when: pxe_boot_result is success

 - name: Attempt to set specific PXE server IP (vendor-dependent)
 ansible.builtin.command:
 cmd: ipmitool -I lanplus -H {{ inventory_hostname }} -U {{ bmc_user }} -P {{ bmc_password }} ipmi boot server set {{ pxe_server_ip }}
 register: pxe_server_result
 ignore_errors: yes # Not all BMCs support setting PXE server IP
 when: pxe_boot_result is success
 notify: Log PXE server IP configuration status
```

handlers:

✕

name: Reboot server

Grok 3

↑

🔖

↺

📝

community.general.ipmi\_power:  
 name: "{{ inventory\_hostname }}"  
 user: "{{ bmc\_user }}"  
 password: "{{ bmc\_password }}"  
 state: boot  
register: reboot\_result  
when: pxe\_boot\_result is success

name: Wait for server to become reachable

ansible.builtin.wait\_for:  
 host: "{{ inventory\_hostname }}"  
 port: 22  
 state: started  
 timeout: "{{ boot\_timeout }}"  
delegate\_to: localhost  
when: reboot\_result is success  
ignore\_errors: yes

name: Log PXE server IP configuration status

ansible.builtin.debug:  
 msg: >-  
 PXE server IP ({{ pxe\_server\_ip }}) configuration  
 {% if pxe\_server\_result is success %}  
 succeeded  
 {% else %}  
 not supported or failed (this is expected for some BMCs)  
 {% endif %}  
when: pxe\_server\_result is defined

## Improvements Over the Previous Version

### • Secure Credential Management:

- Credentials are stored in a separate `vars/bmc_credentials.yml` file, ideally encrypted with Ansible Vault for security.
- Example `vars/bmc_credentials.yml`:

```
bmc_user: "admin"
bmc_password: "password"
yaml
```

### • Encrypt with Ansible Vault:

```
ansible-vault encrypt vars/bmc_credentials.yml
bash
```

Variable Validation:

✕

Grok 3

↑

🔖

↺

📝

Added a `pre_tasks` section to validate required variables (`bmc_user`, `bmc_password`, `bmc_interface`, `pxe_server_ip`) before execution, preventing runtime errors.

• Modular Handlers:

Moved reboot and wait tasks to handlers, triggered only when PXE configuration succeeds, improving playbook flow and reusability.

Added a handler to log the status of the PXE server IP configuration attempt.

• Enhanced Error Handling:

Added a task to check BMC connectivity before attempting configuration, ensuring the BMC is reachable.

Improved retry logic with configurable `ipmi_retries` and `ipmi_delay` variables for all IPMI tasks.

More precise failure condition in the verification task (checks for "PXE" in output).

• Streamlined Execution:

Used `run_once: true` for the IPMI tool installation to avoid redundant checks on the control node.

Simplified the playbook structure by grouping related tasks and using handlers for post-configuration actions.

• Better Feedback:

The PXE server IP configuration task now logs a clear success or failure message via the `debug` handler, improving transparency for vendor-specific limitations.

Verification task provides clearer output for debugging.

• Portability:

Variables like `boot_timeout`, `ipmi_retries`, and `ipmi_delay` are parameterized for easy customization.

The playbook is more adaptable to different BMC vendors by gracefully handling unsupported features (e.g., PXE server IP).

## Prerequisites

### • Ansible Setup:

- Install Ansible and the `community.general` collection:

```
ansible-galaxy collection install community.general
bash
```

- Ensure `ipmitool` is installed on the control node (handled by the playbook).

• **Inventory File:**

✕ Create an inventory file (e.g., `inventory.yml`)



```
bmc_servers:
 hosts:
 bmc1:
 ansible_host: 192.168.1.100
 bmc2:
 ansible_host: 192.168.1.101
```

yaml

Replace `192.168.1.100` and `192.168.1.101` with your BMC IP addresses.



• **Credential File:**

Create `vars/bmc_credentials.yml` with BMC credentials and encrypt it:

```
ansible-vault create vars/bmc_credentials.yml
```

```
bash
```



• **Network Configuration:**

- Ensure the BMC is accessible via IPMI over the network.
- The PXE server at `10.123.232.180` must run DHCP/TFTP services configured to serve boot images.
- The specified interface (`eth0`) must be connected and configured for PXE.

• **BMC Compatibility:**

- The BMC must support IPMI and PXE boot configuration.
- Setting a specific PXE server IP is vendor-specific (e.g., supported by some Dell iDRAC or HPE iLO versions). Check your BMC documentation.

**How to Run the Playbook**

- Save the playbook as `enable_pxe_boot.yml`.
- Set up the inventory file with BMC IP addresses.
- Create and encrypt `vars/bmc_credentials.yml` with BMC credentials.
- Run the playbook, providing the vault password if encrypted:

```
bash
```

```
ansible-playbook -i inventory.yml enable_pxe_boot.yml --ask-vault-pass
```



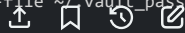
Alternatively, use a vault password file:

```
bash
```



✕ ansible-playbook -i inventory.yml enable\_pxe\_boot.yml --vault-password-file ~/vault\_pass.1

Grok 3



**Notes and Limitations**

- **PXE Server IP:** Most BMCs do not support specifying a PXE server IP directly. The playbook attempts this with a raw IPMI command, but if unsupported, the server relies on DHCP to provide the PXE server (ensure DHCP points to `10.123.232.180`).
- **Interface Selection:** The `lan_interface` parameter assumes the BMC supports selecting a specific NIC. If not, the default interface is used.
- **Vendor Variability:** IPMI commands vary by vendor (e.g., Dell iDRAC, HPE iLO, Supermicro BMC). Test on a single server first.
- **Security:** Always use Ansible Vault for credentials. Avoid plaintext credentials in production.
- **Reboot Behavior:** The playbook reboots the server to apply settings. Ensure this is acceptable in your environment.

**Troubleshooting**

- **BMC Unreachable:** Verify BMC IP, credentials, and network connectivity. Check `bmc_status` task output.
- **PXE Boot Fails:** Confirm the PXE server (`10.123.232.180`) is operational and DHCP/TFTP is correctly configured.
- **Interface Ignored:** Ensure the BMC supports interface selection and `eth0` is valid.
- **IPMI Command Errors:** Update BMC firmware or consult vendor documentation for supported raw commands.
- **Vault Issues:** Ensure the correct vault password or file is provided during execution.

Explain IPMI commands

BMC firmware updates

more concise

**Further Customization**

Ask anything



DeepSearch



Think



Edit Image





Grok 3 