VectraOps

Knowledge Library

TUTORIAL

# How to Configure UFW Firewall Rules on Ubuntu Server

Configure UFW firewall rules on Ubuntu Server with a safe, repeatable workflow: verify prerequisites, allow required services, validate access, and review common mistakes before production use.

**Operating Systems - July 04, 2026**

## Why this matters

The practical problem is simple: a server is online before its network exposure is fully controlled. Without a clear firewall policy, services may be reachable from networks that should never see them, or you may lock yourself out while trying to secure the host.

This tutorial shows how to configure UFW firewall rules on Ubuntu Server in a safe, operationally useful way. By the end, you will know how to verify that UFW is appropriate for your host, enable it without cutting off management access, open only the services you intend to expose, and validate that the resulting rule set behaves as expected before production use.

## What you will build

You will end with a host firewall policy that:

- starts from a deny-by-default inbound posture
- preserves SSH or other remote management access
- allows only the services you explicitly approve
- is validated from both the server side and a remote client perspective

- can be reviewed and adjusted without guesswork

## Prerequisites and safety checks

## Goal

Confirm that UFW is the right control point for this server and that you can safely change firewall state without losing access.

## Action

Before making any changes, verify the following:

- You have console access, out-of-band access, or another recovery path in case remote access is interrupted.
- You know which services must remain reachable, especially SSH on a non-default port if applicable.
- You understand whether the server is behind a cloud security group, network ACL, or upstream firewall that may also filter traffic.
- You are not relying on UFW as the only network control if a separate platform firewall already enforces policy.

Check whether UFW is installed and active:

If the command is not found, install it:

## Expected output

You should be able to determine whether UFW is present, whether it is active, and whether another layer of filtering may exist upstream.

## Validation

- sudo ufw status verbose returns a status such as inactive or active.
- You can name the remote management port and any application ports that must remain open.

## Common failure

A common mistake is enabling a firewall before allowing SSH or another management channel. If remote access is through a non-standard port or a bastion host, verify that path first.

## Stop-here-if warning

Stop here if you do not have a guaranteed recovery path. Enabling a deny-by-default firewall without confirmed access can lock you out of the server.

## Decide the rule set you actually need

## Goal

Define the minimum traffic the server should accept so you do not create overly broad rules.

## Action

List the services you intend to expose, such as:

- SSH for administration

- HTTP and HTTPS for a web server
- a database port only if the server must accept remote database connections
- a custom application port for a specific internal client network

If the service should be reachable only from a specific subnet, plan to restrict the rule by source address rather than allowing it from anywhere.

For example, if SSH should be reachable only from an admin subnet, note that subnet before applying the rule.

## Expected output

You have a short, explicit list of required inbound services and the source networks that should be allowed.

## Validation

A good rule set can be explained in one sentence, such as: ?Allow SSH from the admin subnet and HTTPS from anywhere; deny all other inbound traffic.?

## Common failure

The most common design error is copying generic allow rules without checking whether the service should really be public. Another error is opening the same service from anywhere when a smaller source range would work.

## Enable UFW safely

## Goal

Turn on UFW without disrupting current access.

## Action

If you are connected over SSH, allow your management access before enabling the firewall.
For standard SSH:

If SSH listens on a custom port, allow that port instead:

Then enable UFW:

## Expected output

UFW becomes active and preserves the port you allowed for management access.

## Validation

Check status with line numbers:

You should see a rule that permits SSH or your chosen management port.

## Common failure

If you enable UFW before allowing remote management, your session may drop and remain inaccessible until you recover through console access.

## Add allow rules for required services

## Goal

Permit only the inbound traffic the server needs.

## Action

Use service names when available, or specify ports and protocols directly.

Examples:

If a service should be reachable only from a specific subnet:

If you need to allow a service only to one interface or host, tighten the destination as appropriate for your environment.

## Expected output

UFW contains explicit allow rules for the services and source ranges you approved.

## Validation

Review the active policy:

You should see the intended service ports and, where relevant, restricted source addresses.

## Common failure

A broad rule such as allow 5432/tcp from any source may be acceptable in a lab but is often too permissive for production. Another common mistake is allowing the wrong protocol, such as opening UDP when the service uses TCP.

## Apply a default deny posture

## Goal

Ensure that only approved inbound connections are accepted.

## Action

Set the default policy to deny incoming traffic and allow outgoing traffic:

If you are building a host that must also restrict outbound traffic, treat that as a separate policy design exercise. Most server setups start with open egress and controlled ingress.

## Expected output

New inbound connections are blocked unless they match an explicit allow rule.

## Validation

Confirm the defaults:

The output should show Default: deny (incoming), allow (outgoing) or equivalent wording.

## Common failure

A frequent operational issue is assuming a service is blocked when it is actually still reachable because a permissive rule exists above the default. UFW evaluates explicit rules before default behavior, so review the full list.

# Control access by source when needed

## Goal

Limit sensitive services to trusted networks instead of exposing them broadly.

## Action

For administration services, use source-based rules wherever practical:

For a bastion-only model, allow SSH only from the bastion host?s IP address:

If you already allowed a broad SSH rule, remove it after verifying the narrower rule works.

## Expected output

Only trusted clients can reach the restricted service.

## Validation

Test from an approved source and from an unapproved source if you have a safe way to do so. The approved connection should succeed; the unapproved connection should fail.

## Common failure

CIDR notation mistakes are common. A /24 is far wider than a single host, and an incorrect subnet can accidentally expose a service or block legitimate administration.

# Review, correct, and remove rules

## Goal

Keep the firewall configuration understandable and free of stale entries.

## Action

List rules with numbers:

If you need to delete a rule, use its number:

If you need to replace a broad rule with a narrower one, delete the original after confirming the replacement is correct.

## Expected output

The rule list matches your intended policy and does not contain duplicate or obsolete entries.

## Validation

After any change, run the numbered status output again and confirm the rule order and content.

## Common failure

Deleting the wrong rule is easy if you do not refresh the list immediately before removal. Rule numbers can change after deletions, so always recheck before modifying.

## Validate that the firewall works as intended

## Goal

Confirm that access is allowed where expected and blocked where expected.

## Action

Validate from the server itself and from a remote client.

On the server, check the active state:

From a remote machine, test the allowed services with the appropriate tool. For example, for SSH:

For HTTPS, use a browser or a client such as curl:

For a port-level check, you can use a simple network test tool available in your environment.

## Expected output

Approved traffic reaches the service, while non-approved traffic is rejected or times out according to your network path.

## Validation

- ufw status verbose shows the expected rule set.

- A remote connection to an allowed service succeeds.
- A connection to a blocked service fails.

## Common failure

Do not assume that a successful service test proves the firewall is correct. An application may respond because the test came from an allowed network, or because another firewall layer opened the path.

## Operational follow-up

## Goal

Keep the firewall policy maintainable after initial deployment.

## Action

Record the rationale for each rule: what service it supports, who may use it, and whether the source is restricted.

Revisit the rule set when:

- a service is decommissioned
- a port changes
- access should be narrowed to a smaller network
- a new host is placed behind a separate upstream firewall

If you use automation or configuration management, capture the intended UFW state there so drift is easier to detect.

## Expected output

The firewall remains aligned with current service exposure instead of becoming an outdated list of exceptions.

## Validation

A periodic review should answer three questions quickly: why is this rule present, who depends on it, and can it be narrowed or removed?

## Common failure

The main operational risk is rule drift. Over time, temporary exceptions often become permanent unless someone actively reviews them.

## Common mistakes to avoid

A few UFW mistakes show up repeatedly in real environments:

- Enabling the firewall before allowing the management port
- Opening services from anywhere when only one subnet needs access
- Forgetting that cloud or perimeter firewalls may also need matching rules
- Leaving obsolete rules in place after an application is retired
- Assuming allow rules protect you from application-level exposure

Treat UFW as one layer of control. It is effective when the rule set is narrow, intentional, and verified from the network path that matters.

## Final takeaway

Configuring UFW firewall rules on Ubuntu Server is not just about turning on a firewall; it is about making the server reachable only where required and nowhere else. The safe workflow is straightforward: confirm you have recovery access, allow management first, define only the services you need, enable a deny-by-default inbound policy, validate the active rules from a remote client, and review the configuration over time so it does not drift out of alignment with production needs.