Forcepoint

K3s Installation

(Forcepoint Data Classification & Forcepoint Data Visibility)

Report

Forcepoint

forcepoint.com

Forcepoint
17 August 2024
Forcepoint Proprietary

Report

Table of Contents

Requirements

We use Kubernetes, an open-source container orchestration system to manage our applications. At the moment the only Kubernetes distribution supported is K3s (click here for the official documentation) by Suse Linux for both on-premise and cloud deployments.

The minimum requirements for the Kubernetes cluster are a **single node** (1 virtual machine) with the following specs:

Requirements	Data Classification	Data Visibility	Enterprise (Data Classification + Data Visibility)
CPU	8 cores	16 cores	20 cores
	NOTE: The CPU must support the instructions **SSE4.1, SSE4.2, AVX, AVX2, FMA**. Only x86_64 architecture is supported. Minimum CPU speed is 2.2 GHz		
Memory	32GB	64GB	80GB
Storage	500GB Min available inodes for ext4: 32M	600GB Min available inodes for ext4: 39M	700GB Min available inodes for ext4: 45M
Storage details	→ Only SSD storage is supported → SWAP must be disabled → `/ root` requires at least 20GB → `/var` requires at least 20GB → `/var/lib/rancher` requires at least 500GB (in case of FDC, use the correct disk space according to the type of deployment shown above). → If neither `/var` nor `/var/lib/rancher` is specifically assigned to a partition you must assign the full 500GB to root → If `/var` is specifically assign to a partition but /var/lib/rancher is not, then you must assign the 500GB to `/var` → If `/var/lib/rancher` is specifically assigning to a partition but /var is not, then you must assign the 500GB to `/var/lib/rancher`		
OS	**Ubuntu 20.04** LTS Server is recommended, other supported operating systems include: → Ubuntu 22.04, 24.04 → RHEL 8.8, 8.10, 9.2 and 9.4 → CentOS 7.9 → Suse Linux 15.3 **NOTE:** Only Server edition versions are supported. No Desktop Environment installed. No other Linux distributions are supported. **IMPORTANT**: Root access to the server is necessary during the deployment process and may also be required for support tickets and troubleshooting. Ensure that the service user account accessing the server and deploying the K3s installer has sudo privileges to access and run the installer as root.		
Firewall	→ Port 443/TCP to be open to allow the clients to access dashboard and API → To download application artifacts (Docker images and binaries), updates, and configuration files, the cluster requires a public internet connection with a minimum download speed of 40 Mbps and an upload speed of 8 Mbps. For a faster initial setup, a download speed of 100		

	Mbps or more is recommended.
K3s version support	1.23, 1.24, 1.26
Other Requirements	→ Domain Name Service (DNS) with public name resolution enabled → Network Time Protocol (NTP) service configured → Internet access to a network-based repository for software update packages → Fixed private IPv4 address → Unique static hostname → QoS on network traffic will impact Discovery & Classification speed significantly → Disable any QoS or prioritise traffic to the GV Server cluster → Microsoft can throttle requests to online services such as OneDrive or SharePoint online → This will impact discovery and classification of OneDrive, SharePoint, Teams etc. When deploying using RHEL / CentOS / Suse: → Disable firewalld, fapolicyd, nm-cloud-setup, crypto.fips_enabled → Refer to the official K3s documentation and Forcepoint Data Classification Powered by Getvisibility Prerequisites for K3s on RHEL/CentOS/Oracle Linux for additional setup. When deploying using Ubuntu: → Disable ufw, systemd-resolved, apparmor

Installation

1. If using proxy, run the following command before using `curl`:

   ```
   export http_proxy="$PROXY_IP"

   export https_proxy="$PROXY_IP"
   ```

2. Before installation, use the following command to see if product requirements are met, using (synergy|focus|dspm|enterprise|ultimate) for **PRODUCT_NAME**.

   ```
   curl -sfL https://assets.master.k3s.getvisibility.com/k3s/k3s.sh  | PRODUCT_NAME=enterprise
   ONLY_PRECHECK=true bash -
   ```

3. Run k3s installer using the following command as root user:

   ```
   curl -sfL https://assets.master.k3s.getvisibility.com/k3s/k3s.sh |
   INSTALL_K3S_VERSION="v1.26.10+k3s1" K3S_KUBECONFIG_MODE="644" sh -s - server --node-
   name=local-01
   ```

   Few more arguments that can be used to customize the execution of the k3s script:

   → `SKIP_PRECHECK=true` to skip the execution of the precheck script while installing k3s service
   → `SKIP_SYSTEM_CHECKS=true` to skip the system hardware checking during precheck
   → `SKIP_NETWORK_CHECKS=true` to skip the system network connectivity checking during precheck

   Example: `curl -sfL https://assets.master.k3s.getvisibility.com/k3s/k3s.sh | INSTALL_K3S_VERSION="v1.26.10+k3s1" K3S_KUBECONFIG_MODE="644" SKIP_PRECHECK=true sh -s - server --node-name=local-01`

4. Run the kubectl registration command:

   NOTE: The command below is just an example, it will not work during deployment. For direct customers, Customer Support Team will provide the registration command, otherwise you should have been provided registration command in the Welcome Email.

   ```
   kubectl apply -f https://....k3s.getvisibility.com/v3/import/dxslsxcf84....yaml
   ```

   IMPORTANT: For security reasons the registration command can be used only a single time, the command becomes invalid after the first use. In case you need to run it again you must contact the support team for a new registration command.

5. Monitor the progress of the installation: `watch -c "kubectl get deployments -A"`

   → The K3s deployment is complete when elements of all the deployments (coredns, local-path-provisioner, metrics-server, traefik and cattle-cluster-agent) show at least "1" as "AVAILABLE"
   → In case of errors, you can inspect the logs of a pod using kubectl logs , e.g. kubectl logs cattle-cluster-agent-d96d648d8-wjvl9 -n cattle-system

K3s support matrix

We do not use Docker as the container runtime. Instead, we use containerd.

See the support matrix: EN-2.6.5SupportMatrix-300422-0116-26.pdf

Why K3s?

Kubernetes has been widely adopted in modern software development as it offers a powerful, portable, and open-source platform that automates the management of containerized applications.

When setting up a Kubernetes environment, it comes in two flavours: vanilla Kubernetes and managed Kubernetes. With vanilla Kubernetes, a software development team must pull the Kubernetes source code binaries, follow the code path, and build the environment on the machine. On the other hand, managed Kubernetes comes pre-compiled and pre-configured with tools that improve features to enhance a certain focus area, such as storage, security, deployment, monitoring, etc. Managed Kubernetes versions are also known as Kubernetes distributions. Some popular Kubernetes distributions are Rancher, Red Hat OpenShift, Mirantis, VMware Tanzu, EKS, GKE and AKS.

Kubernetes distributions can have different components that may cause applications that work in one distribution to not necessarily work or even crash into another. Some of the most important components that differ between distributions are:

→ **Container Runtime**: The container runtime is the software that is responsible for running containers. Each Kubernetes Distribution may offer support for different Container Runtimes. Some popular container runtimes include Docker, CRI-O, Apache Mesos, CoreOS, rkt, Canonical LXC and frakti, among others.
→ **Storage**: Storage is important for Kubernetes applications as it offers a way to persist this data. Kubernetes' Container Storage Interface (CSI) allows third-party vendors to easily create storage solutions for containerized applications. Some Kubernetes Distributions build their own storage solutions while others integrate with existing third-party solutions. Popular storage solutions for Kubernetes include Amazon ElasticBlock Storage (EBS), GlusterFS, Portworx, Rook, OpenEBS, among others.
→ **Networking**: Kubernetes applications are typically broken down into container-based microservices which are hosted in different PODs, running in different machines. Networking implementations allow for the seamless communication and interaction between different containerized components. Networking in Kubernetes is a herculean task, and each distribution may rely on a networking solution to facilitate communication between pods, services, and the internet. Popular networking implementations include Flannel, Weave Net, Calico and Canal, among others.

To offer our customers a better and more seamless experience while configuring, running, upgrading, and troubleshooting our

products while also avoiding compatibility issues between different distributions we decided to officially support **ONLY ONE** Kubernetes distribution: **K3s**. The main reasons for choosing K3s are:

a) **Costs** — K3s is 100% open source and there is no need to pay for any expensive licenses.

b) **Less setup overhead** — a lot of time is saved when setting up a new environment because you do not need to go through a lengthy process of acquiring extra licenses based on how many CPU cores you have. Also, K3s can be installed using only one command.

c) **It supports many Linux distros** — K3s supports popular Linux distributions including open-source ones, it can also run both on-premise and in the cloud (AWS, Azure, GCP).

d) **It is fast and lightweight** — K3s is packaged as a single <100MB binary and its lightweight architecture makes it faster than stock Kubernetes for the workloads that it runs.

e) **Easy to update** — Thanks to its reduced dependencies.

f) **Batteries included** — CRI, CNI, service load balancer, and ingress controller are included.

g) **Smaller attack surface** — Thanks to its small size and reduced number of dependencies.

h) **Certified** — K3s is an official CNCF project that delivers a powerful certified Kubernetes distribution.

i) **Flexible** — you can run K3s using single-node or multi-node cluster setup.

Network settings

Your network should be configured to allow the following public urls to be accessible over port **443** (HTTPS) and HTTPS traffic is **bypassed** (NOT intercepted):

1. `https://assets.master.k3s.getvisibility.com (Custom K3s installation files)`
2. `https://images.master.k3s.getvisibility.com (Private Docker registry)`
3. `https://charts.master.k3s.getvisibility.com (Private Helm registry)`
4. `https://prod-eu-west-1-starport-layer-bucket.s3.eu-west-1.amazonaws.com (Docker registry AWS CDN)`
5. `https://rpm.rancher.io (Rancher RPM repo for configuring SELinux packages on RHEL or CentOS)`
6. `https://api.master.k3s.getvisibility.com (Private API server)`
7. `https://rancher.master.k3s.getvisibility.com (Rancher management server)`
8. `https://rancher.$RESELLER_NAME.k3s.getvisibility.com (Rancher management server, where $RESELLER_NAME is Getvisibility for direct customers)`

For more details, see the guide Forcepoint Data Classification Powered by Getvisibility Configuring Rancher and Fleet Agent to Run Behind a HTTP Proxy.

Rancher might be trying to reach to git.rancher.io since it is a default hard-coded repository, but we have our own private repo with all our charts. So, it is ok to block it as we cannot disable it.

Forcepoint

forcepoint.com/contact

About Forcepoint

Forcepoint is the leading user and data protection cybersecurity company, entrusted to safeguard organizations while driving digital transformation and growth. Forcepoint's humanly-attuned solutions adapt in real-time to how people interact with data, providing secure access while enabling employees to create value. Based in Austin, Texas, Forcepoint creates safe, trusted environments for thousands of customers worldwide.

[K3s Installation (Forcepoint Data Classification & Forcepoint Data Visibility)] [Status]17 Aug. 24