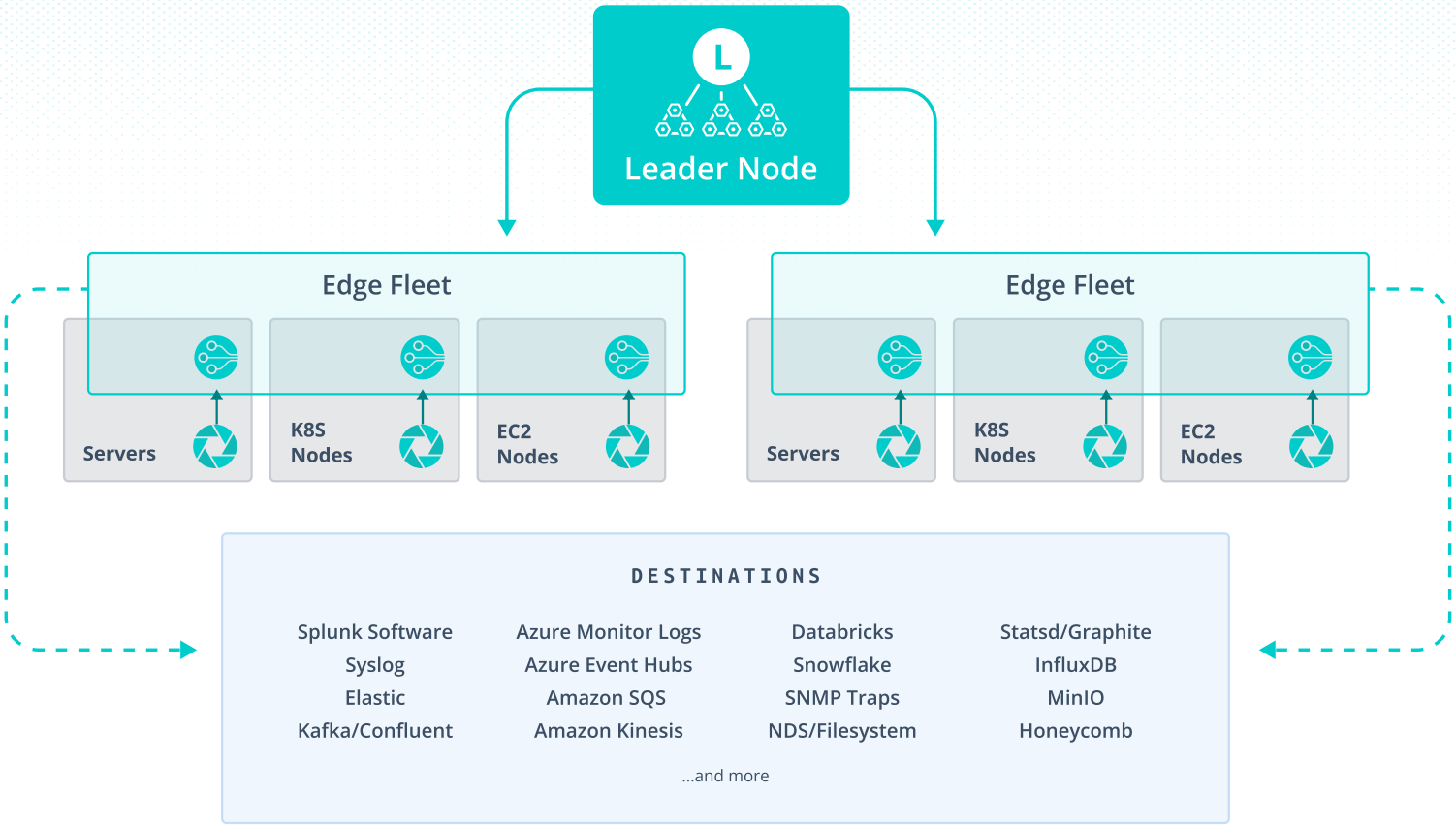
Tip Sheet: Cribl Edge™



Basic Concepts

WHAT IS CRIBL EDGE?

Cribl Edge is an intelligent, vendor-neutral agent built for the scale and complexity of modern architectures. Designed to simplify telemetry collection, Cribl Edge is an easy-to-use agent that automatically discovers and collects data from Windows, Linux, and Kubernetes environments. Centrally managed, configured, and version-controlled, Cribl Edge streamlines agent management to empower teams to operate efficiently at scale.

SOURCES

Seeing as Cribl Edge will be deployed at, well, the edge, we have included some unique Sources that ease the workload of users shipping data to where they want it. Along with some shared sources you've already seen in Cribl Stream, Cribl Edge also provides easy access to: Kubernetes (metrics,logs, events), Linux Journal Files, Linux System metrics, Prometheus Scraper, Windows Event Logs, Windows System metrics, and many more!

DESTINATIONS

All destinations included in Cribl Stream are also available on Cribl Edge, meaning you can send data anywhere you want it. However, given that Cribl Edge will most likely be running on endpoints that are already loaded with work, we most often send our edge data to a Cribl Stream deployment for further processing prior to a final destination. As such, Cribl Edge is primed to ship data to Cribl Stream using the Cribl TCP or Cribl HTTP destinations. Using these outputs ensures that your data ingest is only counted once against your Cribl license. Nice!

Basic Concepts (cont.)

PIPELINES

Most use cases can utilize a **passthru** pipeline or a **Pack** to get the desired results. Sometimes, however, you need to transform your data right then and there. Worry not, Cribl Edge maintains the ability to utilize the pipelines you know and trust from Cribl Stream, including all the same **Functions**.

FUNCTIONS

A Function is a piece of JavaScript code that executes on an event, and it encapsulates the smallest amount of processing that can happen to that event. E.g., a **Function** can replace the term `foo` with `bar` on each event. Another one can hash `bar`, and yet another can add a field, say, `dc=jfk-42` any event from `host us-nyc-42.cribl.io.`

PACKS

Packs enable Cribl Edge administrators and developers to pack up and share complex configurations and workflows across multiple Worker Groups, or across organizations. Packs can contain everything between a Source and a Destination: Routes (Pack-level), Pipelines (Pack-level), Functions (built-in), Sample data files, Knowledge Objects (Lookups, Parsers, Global Variables, Grok Patterns, and Schemas). Wherever you can reference a Pipeline, you can specify a Pack!

EXPLORE

Rather than SSHing to a host and poking around, Cribl Edge provides administrators with a way to **Explore** the host machine right in the UI. Administrators can **Explore**: Running Processes, Containers, Files, and System State. What's more, Cribl Edge will automatically monitor files that are open for writing by current running processes. No more guessing where developers are writing their logs to disk.

COLLECT (QUICKCONNECT)

Speed up shipping of important data through Cribl Edge's use of QuickConnect: a visual rapid-development UI. With it, you can connect Cribl Edge inputs **(Sources)** to outputs **(Destinations)** through simple drag-and-drop. You can then insert **Pipelines** or **Packs** into the connections, to take advantage of Cribl Edge's full range of datatransformation **Functions**. Or you can omit these processing stages entirely, to send incoming data directly to **Destinations** – with minimal configuration fuss. developers are writing their logs to disk.

FLEETS

In a distributed environment, Edge Nodes can be managed centrally by a single Leader Node, which is responsible for keeping configurations in sync, tracking Edge node status and monitoring metrics. Cribl Edge also utilizes **Subfleets.** A **Subfleet** groups and manages Edge Nodes that share the same configuration. Each Subfleet inherits configurations from its parent Fleet. Updating and deploying parent-level configurations applies the changes to the Subfleets. The Subfleets then deploy the configuration changes to their Edge Nodes.

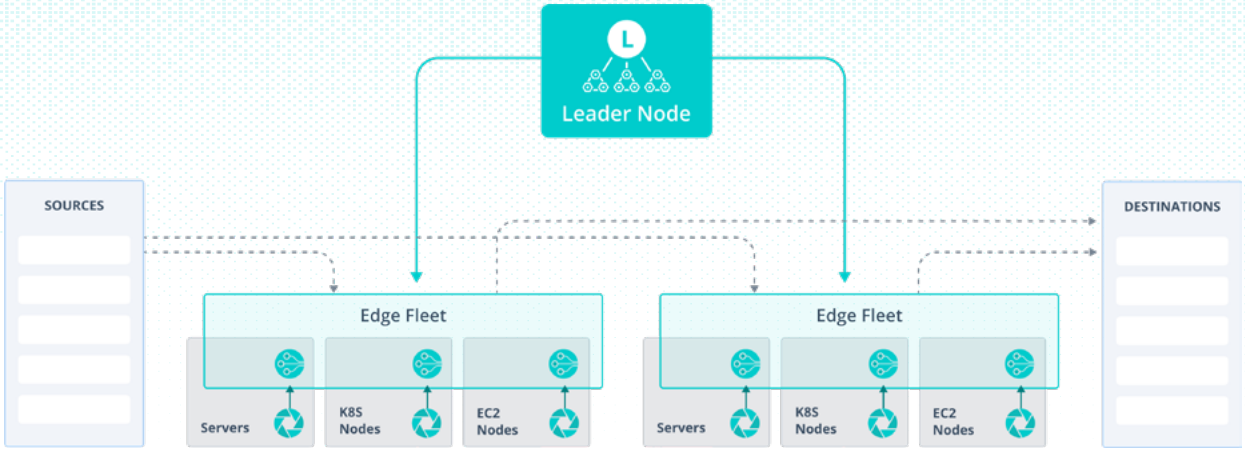
PLATFORMS

You need data from many places that aren't always easy to get to. Cribl Edge supports multiple platforms of deployment to best suit your needs of gathering data: **Docker, Kubernetes, Linux, and Windows.** Each platform has its own straightforward deployment script as well. The script can be found on your leader node in **Manage > Fleets > $FLEET_NAME > Add/Update Edge Node**.

CRIBL.CLOUD

The fast alternative to downloading and self-hosting Cribl software is to launch Cribl. Cloud. This SaaS version, whether free or paid, places the Leader in Cribl.Cloud, where Cribl assumes responsibility for managing the infrastructure.

Fleets



Cribl Edge enables you to flexibly group Edge Nodes into logical Fleets and Subfleets, organized as you like. However, Fleet design is extremely important for the health of the Leader. In this guide, we address "better practices" for designing your Fleets and Subfleets.

DESIGNING YOUR FLEET HIERARCHY

In more complex environments, layering configurations can help you to manage a large number of Edge Nodes that share common yet differentiated configurations. There are several ways you can build your hierarchy:

- **Geographical:** If your organization collects different data categories in different regions, consider this option.
- **Data center**: This is a good option if you maintain data centers in different regions.
- **Server function** (web server, database, etc.): Another good way to organize your Fleets can be based on server functions.
- **OS-based**: If you are running an environment with both Linux and Windows machines, consider separating them into different Fleets per OS.

NAMING YOUR FLEETS

What's in a name, you ask? Well, a whole lot when it comes to Fleets. Consider naming your Fleets and Subfleets in a way that helps you recognize the configurations at a glance. Naming your Fleet with a logical `<parent-fleet>-<subfleet1>-<subfleet2>` will easily provide you with context whether you are viewing at the Fleet, Subfleet, or Edge Node level. It can also help identify Edge Nodes that are out of place or mapped incorrectly.

Common Deployments & Use Cases

DEPLOYMENT PLANNING

Key points to factor in when planning an Edge deployment:

- Amount of data to be collected at the endpoint: how many MB/s or GB/day?
- Amount of processing: are there a lot of transformations, regex extractions, parsing functions, field obfuscations, etc.?
- Routing and/or cloning: Is most data going to a single destination, or is it being cloned and routed to multiple places?
- Deploying onto servers with no internet access: If you plan to deploy Edge Nodes on air-gapped on-prem servers (i.e., servers with no internet access), you must ensure that every Edge Node can communicate with a Leader that is on-prem (i.e., not in Cribl.Cloud).

SYSTEM REQUIREMENTS

WEdge Nodes should have sufficient CPU, RAM, network, and storage capacity to handle your specific workload. It's very important to test this before deploying to production.

REQUIREMENT TYPE	REQUIREMENTS DETAILS
MINIMUM: EDGE NODES	**OS:** Linux: RedHat, CentOS, Ubuntu, AWS Linux, SUSE, Windows (64bit) **System:** ~1 CPU for edge processing, 512MB RAM, 5GB of free disk space (more if persistent queuing is enabled on Workers)
RECOMMENDED: LEADER NODE	**OS:** Linux: RedHat, CentOS, Ubuntu, AWS Linux, SUSE (64bit) **System:** 4 physical cores, +8GB RAM, 5GB free disk space

PERFORMANCE CONSIDERATIONS

Edge Nodes are limited to a single process, and will consume in the very worst case 1 CPU. As with most data-collection and -processing applications, Cribl Edge's expected resource utilization will be proportional to the data volume and type of processing. For instance, a Function that adds a static field on an event will perform faster than one that applies a regex tofind and replace a string.

- Processing performance is proportional to CPU clock speed.
- All processing happens in-memory.
- Processing does not require significant disk allocation.

Thinking about your planned Edge deployment as a whole, it's critical to consider cardinality: how many Fleets, of what size, will send metrics to a given Leader. In a very low-cardinality deployment, a Leader Node might aggregate metrics from one small Fleet; in a very high-cardinality deployment, a Leader Node might aggregate metrics from many large Fleets.

BOOTSTRAPS

Cribl Edge focuses on ease of use / administration /deployment. Therefore, we've included a straightforward way to bring up new Edge nodes once your leader has been configured and your Fleets planned out: Bootstrap Scripts. Below is a quick look at procuring the script and some examples.

PLATFORM	SAMPLE BOOTSTRAP SCRIPT
LINUX	`curl 'https://<org>.cribl.cloud/init/install-edge.sh?group=<token ⩾ $token&user=cribl&install_dir=/opt/cribl' | bash -`
WINDOWS	`msiexec /i https://cdn.cribl.io/dl/4.2.0/cribl-4.2.0-365-win32-x64.msi /qn MODE=mode-managed-edge HOSTNAME=clint-dritan-ledion-nczyjx0.cribl.cloud PORT=4200 FLEET=gymtanlaundry AUTH=$token TLS=true USERNAME="Administrator"APPLICATIONROOTDIRECTORY="C:\Program Files\Cribl\"`
DOCKER	`docker run -d --privileged -e "CRIBL_DIST_MASTER_URL=tls://$token@clint-dritanledion-nczyjx0.cribl.cloud:4200?group=gymtanlaundry" -e "CRIBL_DISTMODE=managed-edge" -e "CRIBL_EDGE=1" -p 9420:9420 -v "/:/hostfs:ro" --restartunless-stopped --name "cribl-edge" cribl/cribl:latest`
KUBERNETES	`helm install --repo "https://criblio.github.io/helm-charts/" --version "^4.2.0" --create-namespace -n "cribl" --set "cribl.leader=tls://$token@clint-dritanledion- nczyjx0.cribl.cloud?group=gymtanlaundry" "cribl-edge" edge`

NOTE: When deploying in Kubernetes, we recommend running Cribl Edge as a DaemonSet. A DaemonSet ensures that all Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from a cluster, said Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

Common Deployments & Use Cases (cont.)

CRIBL EDGE TO CRIBL STREAM

Cribl Edge automatically discovers logs, metrics, application data, etc. – in real time – from your configured endpoints, and delivers them to Cribl Stream or any supported destination. Meanwhile, Cribl Stream can help collect, reduce, enrich, transform, and route data from Cribl Edge to any destination. And using a Cribl TCP/HTTP Source, you can collect and route data from Edge Nodes to Stream Worker Nodes connected to the same Leader, without consuming additional license. A quick overview of steps looks like this:

1. Configure a Cribl TCP/HTTP Source on Cribl Stream
2. Configure required Sources on Cribl Edge
3. Configure a Cribl TCP/HTTP Destination on Cribl Edge
4. Create a Route from your Cribl Edge Sources to your Cribl TCP/HTTP Destination (This can be done either in Collect or in More > Data Routes)
5. Confirm data flow

EXPLORE

Processes
Explore a list of all the processes running on any Edge Node. Every entry comes with basic information on the process, including CPU, Memory, and I/O graphs. Additionally, Edge provides a tabular view of processes'

information out of `/proc`. This makes trouble shooting easier by eliminating the need to SSH to machines. Cribl Edge also provides active listening ports for running processes, again in a tabular view, of all active **Listening, Inbound,** and **Outbound** connections.

Containers
Cribl Edge also provides a list of all the running containers and container metrics on an Edge Node including information about images, volumes, status, ports, etc. Cribl Edge 4.1.x and later supports both Docker and `containerd` runtimes. Cribl Edge also supplies a view into container logs.

Files
As if by magic, all the log files being actively written to by running applications will be auto-discovered and displayed by Cribl Edge. You can also specify a list of directories and files to actively monitor. Files can be discovered with the following options:

Inspect
For each file listed in this section, an **Inspect File** tab is populated with file metadata, including: permissions, file size, user, and modified date. If the file appears suspicious, it can be checked against **VirusTotal** or **OpSwat** to see if the file is flagged as compromised.

Monitor
Each file entry also includes a **Monitor** button to automatically configure a **File Monitor** Source to generate events from the file's lines or records. The **Monitor** feature automatically pre-populates the modal with the following settings configured on the **Files** tab: **Discovery mode, Search path, Max depth, Filename allowlist**

Cribl Edge on Windows

For holistic support for Windows observability allowing comprehensive data collection across your Windows servers to provide reliable monitoring and analysis, look no further. You can run Cribl Edge on your Windows servers (2016, 2019, or 2022) to collect observability data (both metrics and logs) and send them to your desired destinations.

LOCATE WINDOWS EVENT LOGS IN THE SERVER

Event logs in Windows Servers are classified into the following broad categories: System, Applications and Services, Security, and Setup.

You can execute the following Powershell command to obtain a particular log's ID. In this example, we are using the * as a shortcut.

```
powershell Get-WinEvent -ListLog Hardware*
```

From the powershell output, the **LogName** field is what you would specify in the Cribl UI.

BONUS SECTION: EXECUTING POWERSHELL COMMANDS

The Exec Source enables you to periodically execute a command and collect its stdout output. The Exec Source command is actually executed from a Windows Command line, not a Powershell command line.

Note: The execution of PowerShell is resource-intensive and should be used with caution. On limited resource systems (e.g., domain controllers) it is best to run Exec with PowerShell carefully as the Source uses the host's CPU. x

The main thing to keep in mind when you configure Cribl's Exec S ource, is that what you enter in the **Command** field is slightly different from a Powershell prompt command.

POWERSHELL PROMPT COMMAND	CRIBL EXEC SOURCE CONFIGURATION

`Get-Process`
 	
`# cd c:\mydi`
`# .\myps-script.ps1`
	`powershell Get-Process`
	`powershell c:\mydir\myps-script.ps1`

Example: exec command for this script: `Get-CimInstance win32_process | select ProcessId,Name,commandline,HandleCount,WorkingSetSize,VirtualSize`

Enter the following into the Exec Source's **Command** field:

```
powershell -command "Get-CimInstance win32_process | select ProcessId, Name, commandline, HandleCount, WorkingSetSize,VirtualSize"
```

And for the newer version of the Powershell and you can enter the following into the Exec Source's **Command** field:

```
powershell "Get-CimInstance win32_process | select ProcessId, Name, commandline, HandleCount, WorkingSetSize,VirtualSize"
```