**Pixie Overview**

Pixie is an open source observability tool for Kubernetes applications. Pixie uses [eBPF](https://docs.pixielabs.ai/about-pixie/pixie-ebpf) to automatically capture telemetry data without the need for manual instrumentation.

Developers can use Pixie to view the high-level state of their cluster (service maps, cluster resources, application traffic) and also drill-down into more detailed views (pod state, flame graphs, individual full body application requests).

Pixie was contributed by [New Relic, Inc.](https://newrelic.com/) to the [Cloud Native Computing Foundation](https://www.cncf.io/) as a sandbox project in June 2021.

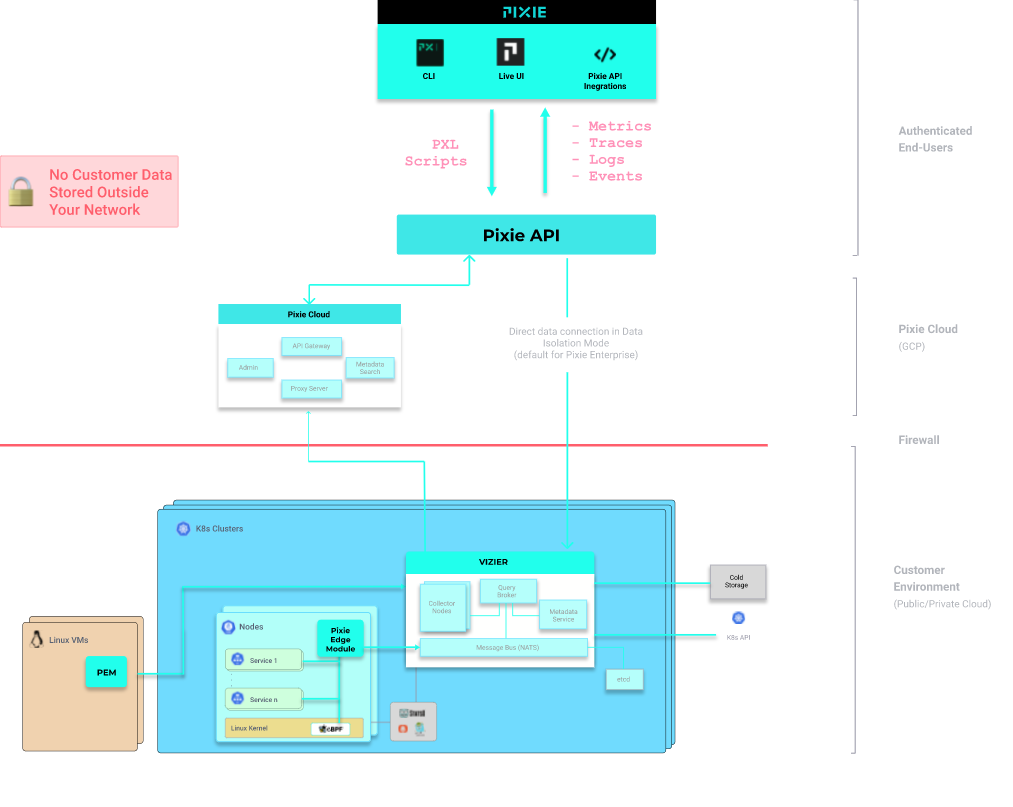
**Features**

* **Auto-telemetry**: Pixie uses eBPF to automatically collect telemetry data such as full-body requests, resource and network metrics, application profiles, and [more](https://docs.pixielabs.ai/about-pixie/data-sources).
* **In-cluster edge compute**: Pixie collects, stores and queries all telemetry data [locally in the cluster](https://docs.pixielabs.ai/about-pixie/faq#where-does-pixie-store-its-data). Pixie uses less than 5% of cluster CPU, and in most cases less than 2%.
* **Scriptability**: [PxL](https://docs.pixielabs.ai/reference/pxl/), Pixie’s flexible Pythonic query language, can be used across Pixie’s UI, CLI, and client APIs. Pixie provides a set of [community scripts](https://github.com/pixie-io/pixie/tree/main/src/pxl_scripts) for common [use cases](https://docs.pixielabs.ai/tutorials/pixie-101).

**Architecture**

The Pixie platform consists of multiple components:

* **Pixie Edge Module (PEM)**: Pixie's agent, installed per node. PEMs use eBPF to collect data, which is stored locally on the node.
* **Vizier**: Pixie’s collector, installed per cluster. Responsible for query execution and managing PEMs.
* **Pixie Cloud**: Used for user management, authentication, and data proxying.
* **Pixie CLI**: Used to deploy Pixie. Can also be used to run queries and manage resources like API keys.
* **Pixie Client API**: Used for programmatic access to Pixie (e.g. integrations, Slackbots, and custom user logic requiring Pixie data as an input)

****

**Data Sources[](https://docs.pixielabs.ai/about-pixie/data-sources/#title)**

Pixie uses [eBPF](https://docs.pixielabs.ai/about-pixie/pixie-ebpf) to automatically instrument Kubernetes applications.

Pixie ships with a set of default data sources, which can also be extended by the user.

**Data Sources[](https://docs.pixielabs.ai/about-pixie/data-sources/#data-sources)**

Pixie automatically collects the following data:

* **Protocol traces**: Full-body messages between the pods of your applications. Tracing currently supports the following [list of protocols](https://docs.pixielabs.ai/about-pixie/data-sources#supported-protocols). For more information, see the [Request Tracing](https://docs.pixielabs.ai/tutorials/pixie-101/request-tracing), [Service Performance](https://docs.pixielabs.ai/tutorials/pixie-101/service-performance), and [Database Query Profiling](https://docs.pixielabs.ai/tutorials/pixie-101/database-query-profiling) tutorials.
* **Resource metrics**: CPU, memory and I/O metrics for your pods. For more information, see the [Infra Health](https://docs.pixielabs.ai/tutorials/pixie-101/infra-health) tutorial.
* **Network metrics**: Network-layer and connection-level RX/TX statistics. For more information, see the [Network Monitoring](https://docs.pixielabs.ai/tutorials/pixie-101/network-monitoring) tutorial.
* **JVM metrics**: JVM memory management metrics for Java applications.
* **Application CPU profiles**: Sampled stack traces from your application. Pixie’s continuous profiler is always running to help identify application performance bottlenecks when you need it. Currently supports compiled languages (Go, Rust, C/C++). For more information, see the [Continuous Application Profiling](https://docs.pixielabs.ai/tutorials/pixie-101/profiler/) tutorial.

Pixie can also be configured by the user to collect [dynamic logs](https://docs.pixielabs.ai/tutorials/custom-data/dynamic-go-logging/) from Go application code and to run [custom BPFTrace scripts](https://docs.pixielabs.ai/tutorials/custom-data/distributed-bpftrace-deployment).

**Supported Protocols**

Pixie automatically traces the following protocols:

| **Protocol** | **Support** | **Notes** |
| --- | --- | --- |
| HTTP | Supported |  |
| HTTP2/gRPC | Partially Supported | Currently only for Golang apps with [debug information](https://docs.pixielabs.ai/reference/admin/debug-info). |
| DNS | Supported |  |
| NATS | Supported | Requires a NATS build with [debug information](https://docs.pixielabs.ai/reference/admin/debug-info). |
| MySQL | Supported |  |
| PostgreSQL | Supported |  |
| Cassandra | Supported |  |
| Redis | Supported |  |
| Kafka | Supported |  |

Additional protocols are under development.

**Encryption Libraries**

Pixie supports tracing of traffic encrypted with the following libraries:

| **Library** | **Notes** |
| --- | --- |
| [OpenSSL](https://www.openssl.org/) | Version 1.1.0 or 1.1.1, dynamically linked. |
| [Go TLS](https://golang.org/pkg/crypto/tls/) | Requires a build with [debug information](https://docs.pixielabs.ai/reference/admin/debug-info). |

# How Pixie uses eBPF

Pixie uses [eBPF](https://www.brendangregg.com/ebpf.html) to drive much of its data collection. This approach enables Pixie to efficiently collect data in a way that requires no user instrumentation (i.e. no code changes, no redeployments).

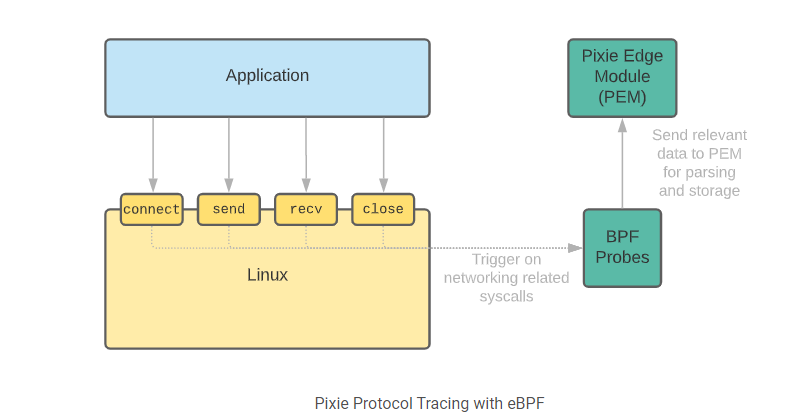
Pixie sets up eBPF probes to trigger on a number of kernel or user-space events. Each time a probe triggers, Pixie collects the data of interest from the event. This enables Pixie to automatically collect monitoring data such as various protocol messages.

The following sections further describe how Pixie uses eBPF in some of its key features.

## Protocol Tracing

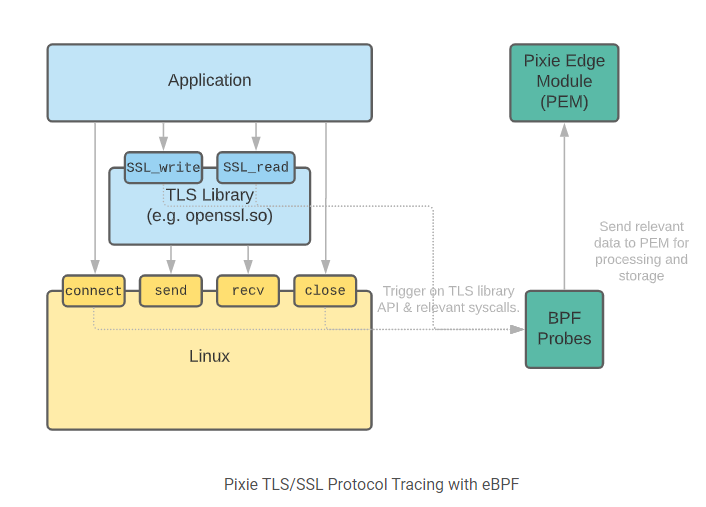
One of the main features of Pixie is its ability to automatically trace protocol messages.

When Pixie is deployed to the nodes in your cluster, it deploys eBPF kernel probes (kprobes) that are set up to trigger on Linux syscalls used for networking. Then, when your application makes network-related syscalls -- such as send() and recv() -- Pixie's eBPF probes snoop the data and send it to the [Pixie Edge Module (PEM)](https://docs.pixielabs.ai/about-pixie/what-is-pixie/#architecture). In the PEM, the data is parsed according the detected protocol and stored for querying.

****

### Tracing TLS/SSL Connections

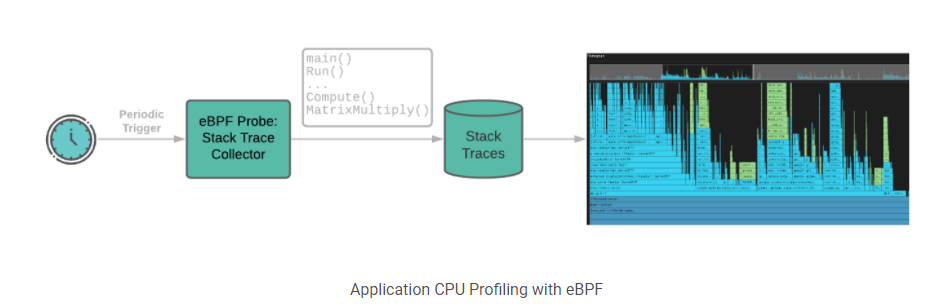
Pixie's SSL tracing works in a similar way as its basic protocol tracing, with one main difference: Instead of using eBPF to snoop the data at send() and recv(), eBPF user-space probes (uprobes) are set up directly on the TLS library's API. This enables Pixie to capture the data before it is encrypted. Pixie supports several [encryption libraries](https://docs.pixielabs.ai/about-pixie/data-sources/#encryption-libraries), including OpenSSL.

****

## Application CPU Profiling

Pixie's [continuous profiler](https://docs.pixielabs.ai/tutorials/pixie-101/profiler) uses eBPF to periodically interrupt the CPU. During this process, the eBPF probe inspects the currently running program and collects a stack trace to record where the program was executing.

This approach to CPU profiling is called a sampling-based profiler. By only triggering at a very low frequency (approximately once every 10 ms), the overhead is negligible. The sampling rate, however, is sufficient to identify the applications that are using the CPU the most, and which parts of the code is typically being executed.



# Installing Pixie

# Requirements

Below are the requirements for deploying Pixie to your Kubernetes (K8s) cluster.

Please refer to the [install guides](https://docs.pixielabs.ai/installing-pixie/install-guides/) for information on how to install Pixie to your K8s cluster.

## Kubernetes

Kubernetes v1.16+ is required.

# Production Environments

| **K8s Environment** | **Support** |
| --- | --- |
| AKS | Supported |
| EKS | Supported (includes support on Bottlerocket AMIs) |
| EKS Fargate | Not Supported ([Fargate does not support eBPF](https://github.com/aws/containers-roadmap/issues/1027)) |
| GKE | Supported |
| OpenShift | Supported |
| Self-hosted | Generally supported, see requirements below including Linux kernel version. |

### Local Development Environments

For local development, we recommend using Minikube with a VM driver (kvm2 on Linux, hyperkit on Mac). Note that Kubernetes environments that run inside a container are not currently supported.

| **K8s Environment** | **Support** |
| --- | --- |
| Docker Desktop | Not supported |
| k0s | Supported |
| k3s | Supported |
| kind | Not Supported |
| minikube with driver=kvm2 | Supported |
| minikube with driver=hyperkit | Supported |
| minikube with driver=docker | Not Supported |
| minikube with driver=none | Not Supported |

## Memory

Memory requirements for your cluster nodes are as follows:

|  | **Minimum** | **Notes** |
| --- | --- | --- |
| Memory | 2GiB | To accommodate application pods, 8GiB+ is recommended. |

## CPU

Pixie requires an x86-64 architecture.

|  | **Support** |
| --- | --- |
| x86-64 | Supported |
| ARM | Not supported |

## Operating System

Pixie runs on Linux nodes only.

|  | **Support** | **Version** |
| --- | --- | --- |
| Linux | Supported | v4.14+ |
| Windows | Not Supported | Not in roadmap |

### Linux Distributions

The following is a list of Linux distributions that have been tested.

|  | **Version** |
| --- | --- |
| Ubuntu | 18.04+ |
| Debian | 10+ |
| RHEL | 8+ |
| COS | 73+ |

Pixie may also work on other distributions.

## Network Traffic

Pixie's Vizier Module sends outgoing HTTPS/2 requests to withpixie.ai:443.

Your cluster's data flows through Pixie's control cloud via a reverse proxy as encrypted traffic without any persistence. This allows users to access data without being in the same VPC/network as the cluster. Pixie offers [end-to-end encryption](https://docs.px.dev/about-pixie/faq#how-does-pixie-secure-its-data) for telemetry data in flight.

# Install Guide

## Prerequisites

* Review Pixie's [requirements](https://docs.pixielabs.ai/installing-pixie/requirements) to make sure that your Kubernetes cluster is supported.
* Determine if you already have [Operator Lifecycle Manager](https://docs.openshift.com/container-platform/4.5/operators/understanding/olm/olm-understanding-olm.html) (OLM) deployed to your cluster, possibly to the default olm namespace. Pixie uses the Kubernetes [Operator pattern](https://kubernetes.io/docs/concepts/extend-kubernetes/operator/) to manage its Vizier, which handles data collection and query execution (see the [Architecture](https://docs.pixielabs.ai/about-pixie/what-is-pixie/#architecture) diagram). The OLM is used to install, update and manage the Vizier Operator.
* Pixie interacts with the Linux kernel to install BPF programs to collect telemetry data. In order to install BPF programs, Pixie [vizier-pem-\*](https://docs.pixielabs.ai/about-pixie/what-is-pixie/#architecture) pods require [privileged access](https://github.com/pixie-io/pixie/blob/main/k8s/vizier/bootstrap/pod_security_policy.yaml).

## 1. Sign up

Visit our [product page](https://work.withpixie.ai/) and sign up.

## 2. Set up a Kubernetes cluster (optional)

If you don't have a Kubernetes cluster available, you can set up Minikube as a local sandbox environment following these [instructions](https://docs.pixielabs.ai/installing-pixie/setting-up-k8s/minikube-setup).

## 3. Install the Pixie CLI

The easiest way to install Pixie's CLI is using the install script:

# Copy and run command to install the Pixie CLI.

bash -c "$(curl -fsSL https://withpixie.ai/install.sh)"

For alternate install options (Docker, Debian package, RPM, direct download of the binary) see the [CLI Install](https://docs.pixielabs.ai/installing-pixie/install-schemes/cli/) page.

## 4. Deploy Pixie 🚀

Pixie's CLI is the fastest and easiest way to deploy Pixie. You can also deploy Pixie using [YAML](https://docs.pixielabs.ai/installing-pixie/install-schemes/yaml) or [Helm](https://docs.pixielabs.ai/installing-pixie/install-schemes/helm). You can use these steps to install Pixie to one or more clusters.

To deploy Pixie using the CLI:

If your cluster already has Operator Lifecycle Manager (OLM) deployed, deploy Pixie using the `--deploy\_olm=false` flag.

# List Pixie deployment options.

px deploy --help

# Deploy the Pixie Platform in your K8s cluster (No OLM present on cluster).

px deploy

# Deploy the Pixie Platform in your K8s cluster (OLM already exists on cluster).

px deploy --deploy\_olm=false

Pixie deploys the following pods to your cluster. Note that the number of vizier-pem pods correlates with the number of nodes in your cluster, so your deployment may contain more PEM pods.

NAMESPACE NAME

olm catalog-operator

olm olm-operator

pl kelvin

pl nats-operator

pl pl-nats-1

pl vizier-certmgr

pl vizier-cloud-connector

pl vizier-metadata

pl vizier-pem

pl vizier-pem

pl vizier-proxy

pl vizier-query-broker

px-operator 77003c9dbf251055f0bb3e36308fe05d818164208a466a15d27acfddeejt7tq

px-operator pixie-operator-index

px-operator vizier-operator

To deploy Pixie to another cluster, change your kubectl config current-context to point to that cluster. Then repeat the same deploy commands shown in this step.

## 5. Invite others to your organization (optional)

Add users to your organization to share access to Pixie Live Views, query running clusters, and deploy new Pixie clusters. For instructions, see the [User Management & Sharing](https://docs.pixielabs.ai/reference/admin/user-mgmt) reference docs.

## 6. Use Pixie

### Deploy a demo microservices app (optional)

Deploy a simple demo app to monitor using Pixie:

# List available demo apps.

px demo list

# Example: deploy Weaveworks' "sock-shop".

px demo deploy px-sock-shop

This demo application takes several minutes to stabilize after deployment.

To check the status of the application's pods, run:

kubectl get pods -n px-sock-shop

### Test out the CLI

Use px live to run a script to demonstrate observability. The http\_data script shows a sample of the HTTP/2 traffic flowing through your cluster.

# List built-in scripts

px scripts list

# Run a script

px live px/http\_data

For more information, checkout our [CLI guide](https://docs.pixielabs.ai/using-pixie/using-cli/).

### Explore the web app

Open [Pixie's Live UI](https://work.withpixie.ai/) in a new tab.

1. After reviewing the hints, click the X in the upper left hand corner of the screen.
2. Select your cluster (you may see other clusters from members of your organization).
3. Now, select a script, e.g. px/cluster or px/http\_data.

For more information, check out our [Live UI guide](https://docs.pixielabs.ai/using-pixie/using-live-ui/).

### Check out the tutorials

Learn how to use Pixie for

* [Network Monitoring](https://docs.pixielabs.ai/tutorials/pixie-101/network-monitoring/)
* [Infra Health](https://docs.pixielabs.ai/tutorials/pixie-101/infra-health/)
* [Service Performance](https://docs.pixielabs.ai/tutorials/pixie-101/service-performance/)
* [Database Query Profiling](https://docs.pixielabs.ai/tutorials/pixie-101/database-query-profiling/)
* [Request Tracing](https://docs.pixielabs.ai/tutorials/pixie-101/request-tracing/)